Product Development Strategies in Evolutionary Acquisition

by

Bobak Ferdowsi

B. S. Aeronautics and Astronautics
University of Washington, 2001

Submitted to the Department of Aeronautics and Astronautics
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Aeronautics and Astronautics

at the
Massachusetts Institute of Technology
September 2003

Signature of
Author……
Department of Aeronautics and Astronautics
August 22, 2003

Certified
by……
Eric Rebentisch
Research Associate, Center for Technology, Policy, and Industrial Development
Thesis Supervisor

Certified
by……
Debbie Nightingale
Professor of the Practice, Department of Aeronautics and Astronautics
Aeronautics and Astronautics Reader

Accepted
by……...
Edward M. Greitzer
H.N. Slater Professor of Aeronautics and Astronautics
Chair, Committee on Graduate Students

Product Development Strategies in Evolutionary Acquisition

by

Bobak Ferdowsi

Submitted to the Department of Aeronautics and Astronautics Engineering
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Aeronautics and Astronautics

ABSTRACT:

Programs are consistently faced with the decision on how to best deliver capabilities to the user. This challenge is magnified by the number of uncertainties and risks the program can expect throughout the product development process—that is the means by which the program gets from perceived need to deliverable solution. The Air Force, as a result of increasing development times for new products, has decided to implement an Evolutionary Acquisition strategy, meaning the development process focuses on delivering incremental capabilities through short increments or spirals.

The question, however, is whether this strategy makes sense across the broad spectrum of Air Force programs. More importantly, how can the Air Force, and aerospace programs in general, decide what product development strategy applies to each program? An extensive literature review yielded a number of relevant questions regarding product development. The hypothesis of this research is that this selection should be based on attributes of the product, the program goals, the uncertainties, and stakeholder involvement.

A number of case studies were performed specifically targeting programs identified as evolutionary. Analysis of these case studies showed that there was a disparity among these programs as to what was meant by Evolutionary Acquisition. These programs were shown to have a mix of strategies, but primarily followed a pattern of creating upgrades to a baseline or platform.

From the case studies a number of key recommendations were made for program managers and policy makers implementing Evolutionary Acquisition, as well as a notional illustration of product development selection. Specifically, the research found that programs with high user requirements uncertainty tended towards more iteration, while programs with significant technical and performance goals had less iteration and greater planning. In programs where rapid delivery was the goal and uncertainties were relatively low, incremental strategies were found to be most applicable. Recommendations included implementing open architecture systems through owning system interfaces, managing stakeholder expectations through system representations and internal change agents, and providing stable funding and contingency funds for rapid change implementation. Additional recommendations were made on implementing testing and logistics in highly iterative programs.

Thesis Supervisor: Eric Rebentisch
Title: Research Associate, Center for Technology, Policy, and Industrial Development

BIOGRAPHICAL NOTE:

Bobak Ferdowsi was born on November 7, 1979, in Philadelphia, Pennsylvania. With his loving parents, he quickly moved to the San Francisco Bay Area where he would stay until the age of 11. In 1991, he moved to Tokyo, Japan, attending the American School In Japan until graduating in 1997, with the exception of a one-year term in Oakland, California at the College Preparatory School. Bobak returned to the United States in the fall of 1997 to enroll in the University of Washington in Seattle.

During his college experience, Bobak toyed with a number of majors before settling on aerospace engineering, a childhood dream of his. At the same time, he worked in the Department of Physics, performing research under Nobel Laureate Dr. Hans Dehmelt and Dr. Warren Nagourney. His professors in the Department of Aeronautics and Astronautics, especially Professor Kuen Lin, would influence his next decision.

Bobak enrolled in the Massachusetts Institute of Technology in the fall of 2001, and joined the Lean Aerospace Initiative. It was here that he learned about a great deal more than the traditional aspects of engineering. The result is a thesis on the Air Force's acquisition strategies, and more importantly, a young man with a vastly greater knowledge of the world of aerospace engineering and an interest in shaping the future of this industry.

Though no one knows for certain what the future holds for Bobak, his aspirations are to pursue a challenging career in aerospace engineering, and eventually, following in the footsteps of his father and of the many who have helped him over the years—become a teacher.

ACKNOWLEDGEMENTS:

If I have grown over the last two years it is because I spent so much time with all of these people who enriched my life.

My advisor, Dr. Eric Rebentisch, for his relentless pursuit of research excellence when I was lazy, but importantly for the great conversations we had that were not related to research. I have learned about more than product development strategies in my time here. Thank you to the Lean Aerospace Initiative: I spent a lot of time with many of the LAI members in our LEAP project, on many Wednesday nights, and a handful of (too) early breakfasts. These include Kirk Bozdogan, Hugh McManus, Earll Murman, Debbie Nightingale, Tom Shields, and Al Haggerty who took me on my first case study. A special thanks to the woman who brought me here and helped shape my research, the late Dr. Joyce Warmkessel.

To my many friends at LAI (and some at SSPARC): Jason Derleth, Lt.Col Chris Forseth, Marc Haddad, Sandra Kassin-Deardorff, Chris Roberts, Adam Ross, Nirav Shah, Dave Stagney, Myles Walton, and Annalisa Weigel. Thanks to Cory Hallam and his soon-to-be wife Sara, for reminding me that there's life outside of MIT. Erisa Hines, my occasional workout partner, and the little sister I never had. Thanks to Alexis Stanke, who guided me through the LAI process, for being a good friend and taking me out when I needed it. To Heidi Davidz and Mike Cluff, my close friends, and the first of many married couples in the last two years, for always being there. Of course a big thank you to Deneen Silviano and Major Ron Jobo, for being the cutest couple, and watching out for me, both in my research and in my life. A special thanks to my surrogate family in Boston, Tim Spaulding and his wife Dori, for taking me in on the holidays, and being great friends.

I would like to thank my family for supporting me in all my endeavors, especially my mother and father, who, despite all our arguments, are the best a boy could hope for.

Thank you to Allison, my girlfriend, for taking such wonderful care of me, and doing whatever she can to cheer me up when I was down.

A special thanks to my good friends here and away, especially David Vener and Wesley Gifford, for putting up with me as a roommate and a friend, going out with me almost every weekend, and keeping my life interesting. Thank you to Parsi Parsinejad, for being my friend since middle school and always making my laugh.

Thanks to all of the participants in my research, for being honest and teaching me a career's worth of lessons in such a short time. Also thanks to the folks at the Acquisition Center for Excellence for giving me an idea for research and helping me out. I would especially like to thank Lt.Col Robert Dare, who made sure I could get in touch with people in the Air Force and providing me with helpful insight on my research. Thanks to Don Blake, the victim of my first case study, for giving me good advice.

There are so many people in these last couple of years that I would like to thank for their help and support, and many more who I am probably forgetting. Thank you all.

TABLE OF CONTENTS

TABLE OF FIGURES

LIST OF TABLES

1 Introduction

Product development is inherently a risk-driven process. Companies take risks in developing new products, or alternatively in not developing new products. This is no different in the aerospace community, in both commercial and military products. Environments are highly competitive and dynamic—commercial products vie for market space, while military products compete for superiority. Both groups are faced with the challenge of strategically developing new products and technologies, or in some cases, understanding when not to develop them.

Due to the nature of aerospace products—their complexity, size, and necessary quality—the decision to engage in new projects is often costly. Many times, especially in the commercial world, new products are often 'betting the company'. This is evidenced by the new Boeing project, sleighed to be the 7E7 Dreamliner, and the Airbus A380, the latter estimated to be a 10 billion dollar undertaking. In the military however, poor development decisions can mean wasted taxpayer money, and more importantly, lost lives.

Many challenges stand in the way of successful product development. The risks involved are significant, and uncertainty is greater in the earlier stages of the project. At the same time however, significant product and process decisions must be made at this time that will affect the product outcome. In addition to this, numerous interfaces, both within the developing organization, and externally to its various stakeholders, are crucial to the process. Furthermore, there is a great deal of complexity within these external

organizations. As a result, the uncertainty associated with new products has driven many different approaches to dealing with these risks.

The uncertainties involved in military programs are of special concern, and though they are not necessarily more complex than commercial systems, subtle differences do exist. For example, the U.S. Military is tasked with maintaining superiority over potential threats, which is analogous to, but not the same as, a company's desire to stay ahead of its competitors. This has typically been achieved through the acquisition of state-of-the-art weapons systems, but this is no longer as easy as it once was. With the end of the Cold War, defense budgets have slowly declined, while the costs of maintaining cutting-edge technologies have increased. The Cold War also implied a clear sense of enemy, and the type of threat to be encountered. Though the war taxed resources, it was significantly easier to predict what technologies were needed to maintain an edge over the enemy.

The challenges faced by the aerospace community came to a head with the events of September 11, 2001. That is not to say that the difficulties in product development did not exist before this time—only that this event publicized product development shortfalls and demonstrated a clear need for improved product development methodologies in the aerospace industry. In particular it demonstrated the dynamic nature of the world today. The commercial world was faced with the fact that fewer people wanted to fly, and the development decisions of major players Boeing and Airbus were called into question. Airline bankruptcies showed that the traditional business models used to justify new products had to be reevaluated. At the same time, like no other event since World War II,

the United States was confronted by the fact that is no longer facing a stable and predictable enemy like the USSR, but rather a new threat, against which traditional strategies and approaches may not be as effective.

Beyond the dynamic world we live in and the evolution of threats, there are many other challenges facing military product development.

One of the significant issues is the increasing cost of military products. Norm Augustine noted in the late 1980s, that without intervention, by 2054 the military budget would afford one aircraft. Significant work has been done to reduce costs through 'Lean' initiatives, a waste-reduction methodology practiced in the Japanese automobile industry for some time. Nevertheless, the majority of these changes have taken place in manufacturing, with changes in supplier networks and product development only recently beginning to emerge. At the same time, the engineering community has become critically aware that design choices determine an estimated 70 to 80 percent of the product cost. The result has been a strong push, like that in the military, to make better design decisions.

A significant part of these decisions involve user feedback. Often times users' first experiences with the aircraft are after the delivery of the product, when changes are not economically feasible. This is magnified by the long cycle times and significant expenditures, so that users are getting products suited to their predicted needs of 10 or more years ago, as shown in Figure 1, and unable to update these products.

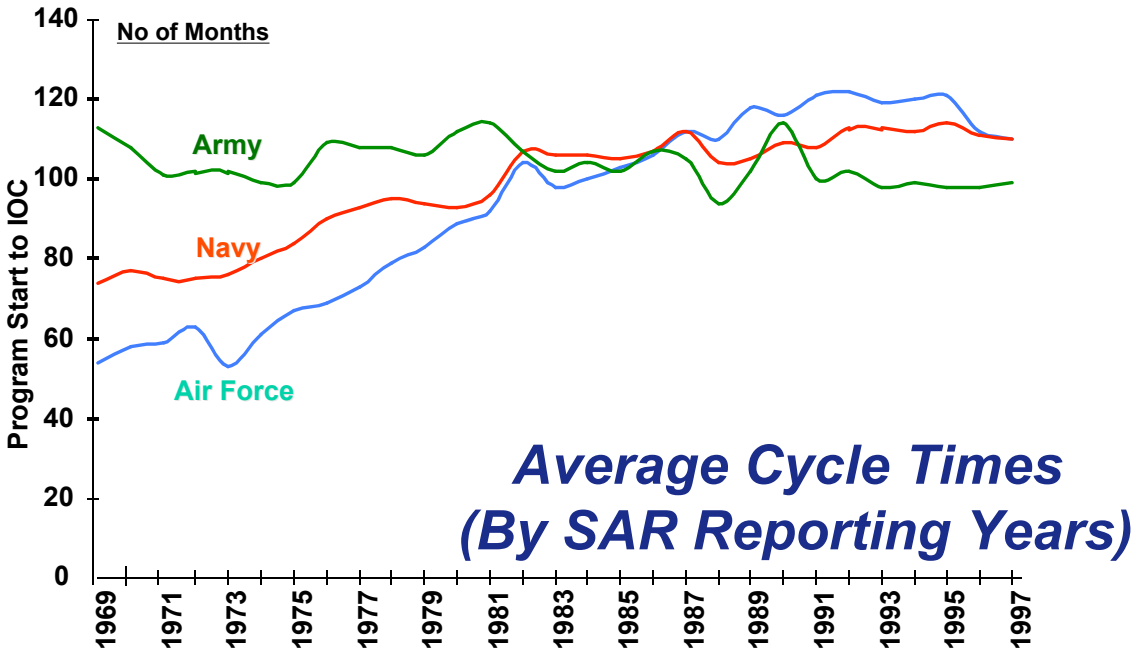


Figure 1: Average Cycle Times

Furthermore, in the current process, changes to the needs often result in schedule slip and budget increases. The military, in particular, faces a significant challenge. In the last several years, there has been a strong push for a new acquisition process to address some of the problems associated with traditional development processes. This essentially means changing the way new products are developed for the military customer. The process is the way in which these products are brought to the customer. For the most part, a development process involves a number of key steps, including formulation of customer needs, a product strategy defining the general way in which those needs can be satisfied, a system design, including detailed designs of all the product components, and testing and evaluation before the product is delivered.

Explicitly, the military is interested in providing the warfighter with capability sooner, as typical development cycle times for aircraft, for example, are on the order of 10 to 12

years (Fine 1998), and for more complex products, such as the F-22, 25 years will have elapsed between concept development and operational capability. In a recent talk Dr. Marvin Sambur, Assistant Secretary of the Air Force for Acquisition, noted, "when it takes so long, it just can't be state-of-the-art" (Paone 2002). Furthermore, as September 11 demonstrated, threats can change significantly in such a timeframe. In addition to this, as a result of the long development times for new products, "large steps in performance are required to justify the need for a new system vs. incremental upgrade," which in turn allows for requirement and specification creep (Wirthlin 2000).

The proposed solution to all this is the new strategy implemented in Air Force acquisition, which emphasizes Evolutionary Acquisition, with a preferred spiral development process. Simply stated, the desire is to deliver partial capability products, or increments, in shorter time, to allow for user feedback and changing needs through greater flexibility. The goal is cycle times of 2 to 4 years, about a quarter of what they are currently. In the spiral process, capability is built up through multiple iterations while allowing for flexibility in design (discussed further in Chapters 2 & 3).

But is that realistic? Does the spiral process adequately address all development projects? Can significantly complex tasks be reduced to such short time scales? Other issues arise, such as the nature of the product, the types of risk likely to be encountered in the development, and so forth.

Hypothesis

The product development process selection is highly complex, and it is likely that no one process will address all the issues associated with product development. In fact, various development strategies address various risks, and the process selected should reflect the risks associated with that program. Furthermore, products themselves introduce various uncertainties, and the strategy should tackle those uncertainties as well. Additionally, the user preferences will drive the process selection as well, due to both the schedule and flexibility of the various processes. In total, the basic presumption for this work is that product development is a marriage between the various aspects of development associated with new products—including stakeholder value, product choices, and development uncertainties. It's hypothesized that a successful marriage between the process and uncertainty will generate greater stakeholder satisfaction.

Key Questions

This thesis investigates a number of questions about product development processes. These can be summarized as:

1. What knowledge is necessary to adequately select a product development strategy?
2. When does Evolutionary Acquisition make sense?
3. How can the acquisition community best implement these various processes, particularly Evolutionary Acquisition?

Deliverables

There are several tasks that this thesis sets out to accomplish and deliver to provide useful knowledge and ability to project management:

1. A framework to evaluate risks, processes, and products, based on the findings of this research as well as historical literature and prior research.
2. A clear understanding of the current state of product development in the military, including the process portfolio and possible lessons learned from product development programs.
3. A tool, based on the findings of the research, to provide project managers with greater insight on the product strategies and processes applicable to that project.

Focus

This thesis will specifically focus on the military acquisition process, and how to effectively address the needs of this community through various product development processes. At the same time, the work will draw extensively from commercial practices and strategies, both in and outside of the aerospace industry.

Summary

The global aerospace community, including the military, faces a great challenge in product development—a push to reduce costs and development times while improving performance and satisfying customer needs. These issues are fundamental to the continued success of the military and commercial aerospace communities. Though the majority of the work done in this thesis will address military issues, both the commercial and military communities experience similar development challenges, and lessons learned and tools can be extracted to commercial programs as well.

Specifically, this research will investigate the synergies between the user needs, product, and process, to show that more effective product development can be achieved through the proper alignment of uncertainties and process. The goal of the work is to provide the acquisition community with an understanding of the current state of affairs, as well as a tool to allow for improved process selection.

The thesis is broken up into three major sections. The first section is a review of the literature studied, followed by a brief look at the research methodology. The second major section is a series of case studies in Evolutionary Acquisition. The final section is an analysis of these case studies concluded by recommendations for program managers and policymakers.

The first section, including Chapters 2 and 3, focuses on the background research in this thesis. Chapter 2 provides an introduction to product development, and looks at aspects of the product and project scope to identify attributes of the system being developed and the strategic goals of the project. Chapter 3 goes into depth about the uncertainties that are typically encountered in a product development process. The bulk of Chapter 3 looks at the variety of theoretical product development strategies, identifying attributes that make them unique and understanding what types of uncertainty each of the processes address. This is concluded by a series of key research questions that emerged from the literature review.

The second section begins with a brief overview in Chapter 4 of the research methodology, specifically focusing on the research design and choices made in the course of exploration. Chapter 5 is an in depth look at each of the case studies, reviewing the program background, acquisition strategy, program uncertainties, key enablers in acquisition, and challenges towards implementing evolutionary strategies.

The final section, beginning with Chapter 6, provides a summary of the case studies and delves into an analysis of the acquisition strategies, followed by a number of interesting relationships between product choices and processes, uncertainties and key approaches to dealing with these, finished by a look at resources in Evolutionary Acquisition. Chapter 7 provides key recommendations for selecting process strategies. After this, recommendations are specifically focused on implementing Evolutionary Acquisition, broken into suggestions for program managers and for policymakers, before concluding with areas for future research.

2 Products and Program Scope

The vast literature focused on product development is both a blessing and a curse. The result of such extensive literature means that there are both a multitude of development strategies and some disagreement as to the definition of these strategies. The distinction of strategies is of utmost importance, as each strategy addresses different uncertainties associated with the product development process. Furthermore, product development has become crucial to success in the marketplace, as companies rely on new products to capture and maintain market share and to revitalize the firm's intellectual capital. This is particularly true in the aerospace industry, where the risk of failure is significant, and companies are often jockeying for multi-million dollar contracts. This chapter will specifically look at two major regions of product development; 1) the attributes of products developed in the aerospace context and 2) the underlying goals of a product development process and the context of a product in a portfolio of systems.

What does it take then to be successful in market? The majority of literature focuses on "innovation" as the key to success. Innovation itself is a challenging topic and has multiple meanings. Webster's Dictionary (1996) defines innovation as:

1. *The act of innovating; introduction of something new, in customs, rites, etc.*
2. *A change effected by innovating; a change in customs; something new, and contrary to established customs, manners, or rites.*

Both of these are highly relevant to industry—companies must both introduce new products and change the way they both develop these products and arrange their organizations. The challenge of product development lies in understanding what a

successful product is before designing and developing such a product. In a sense, companies must identify the target before going after it. Ulrich and Eppinger (2000) isolate five factors by which to judge product success:

1. *Product quality – whether the user needs are satisfied, and the product is reliable.*
2. *Product cost – the cost of the product, both to setup manufacturing and to produce.*
3. *Development time – the time required for the project team to develop the product, and whether the team is agile enough to respond to changes.*
4. *Development cost – the expenditure required to develop the product.*
5. *Development capability – the usefulness of the product development process in terms of generating abilities for future product development.*

Of particular interest, as one of the key questions in this thesis, is the means by which the process used in product development affects the outcome of the product. Much of the literature available realizes the benefits of reduced time and resources in bringing a product to market through the proper selection of development process (Eisenhardt & Tabrizi 1995). This is of significant importance to the aerospace industry where resources are at a premium, and specifically to the military complex, which, as stated before, faces the problem of both increasing acquisition costs and average cycle time increases on the order of 100 percent in the last thirty years. In addition, the benefits of short cycle times is necessary in the military in order to deal with a dynamic threat environment. Accordingly, this strategic focus on product development has spurned several new initiatives, specifically Evolutionary Acquisition and spiral development,

which seek to provide the warfighter with tangible, useful product in a short amount of time, and allow for feedback and evaluation in the ongoing development process.

Product development strategies break down into a number of different attributes, each important in the strategy and selection of process. Accordingly it is necessary to explore the multitude of categories in which product development applies, and to scope the problem to a feasible task.

Essentially, it is possible to break the development literature into three macro categories, though there is a high degree of correlation and integration amongst them. These three include organizational innovation, product innovation, and process innovation. Due to the nature of the recent focus of the military acquisition community, specifically in Evolutionary Acquisition, the bulk of this research focuses on product and process innovation, and specifically on synergies that exist within the two. At the same time, the author realizes the essential need for organizational change, and that the selective use of resources within the organization affects the outcome of product development.

Although this thesis investigates the product-process synergies, the organizational implications are essential to the picture. This is not a one-way street—the organization affects the process and product, but at the same time, the process and product influence the organization. In particular, new product creation is an essential part of the company's ability to adapt and evolve (Dougherty 1992). This is of importance to the military, where the mission threat is dynamic and evolving.

One potentially useful representation of the product development process comes from an extensive literature review by Brown and Eisenhardt (1995). This review focused on the more traditional organizational literature, but several aspects are essential to the realms of product and process innovation. Figure 2 shows the Brown and Eisenhardt illustrated model of product development. Of note are the bold lines and capital letters, which indicate verified and robust relationships. The remaining points are put forth as relevant, but not necessarily primary factors in product success.

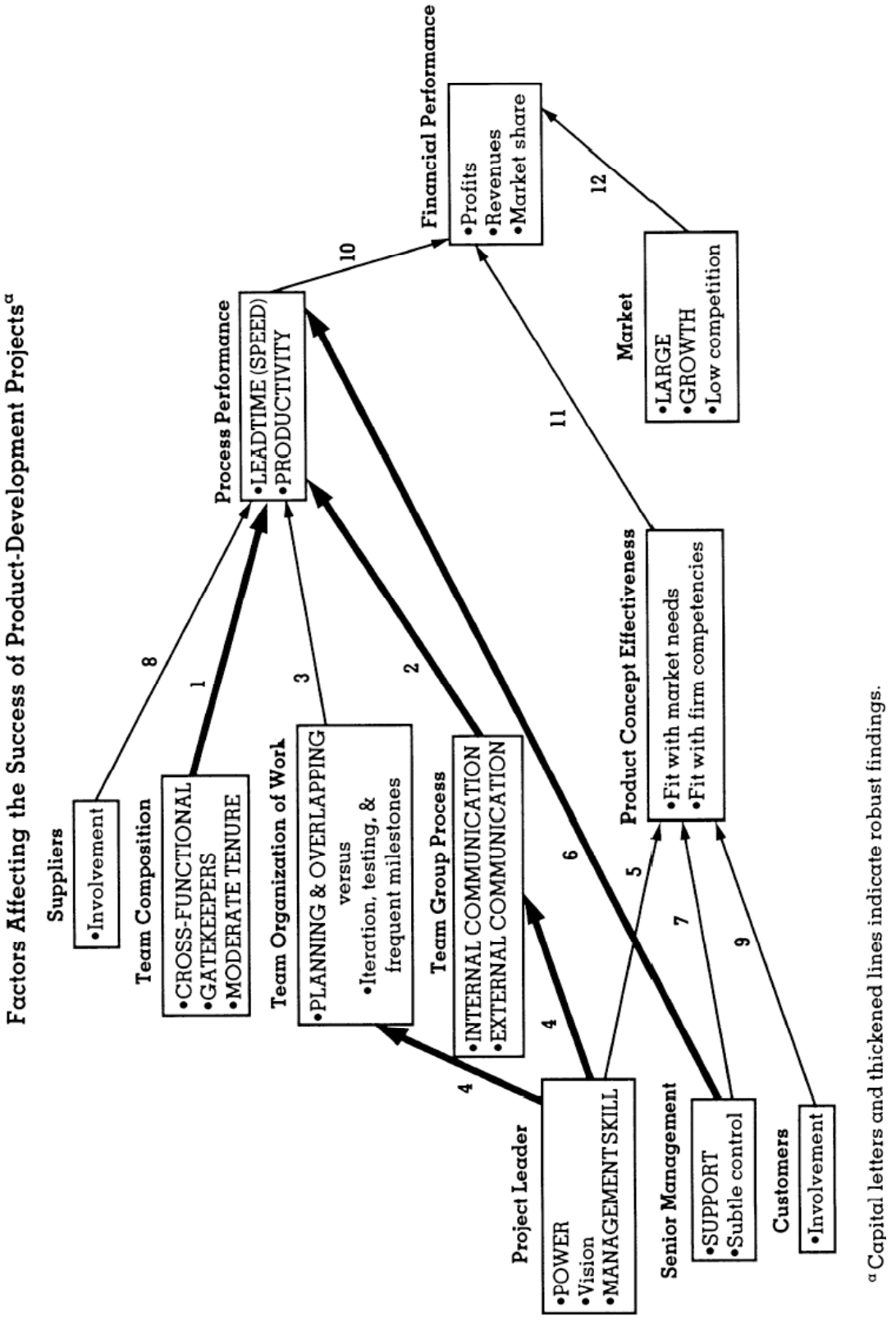


Figure 2: Brown and Eisenhardt (1995) model of product development

In particular, the figure highlights several factors that appear to have the greatest relevance to product development, including:

- External communication
- Organization of work
- Customer involvement
- Product concept fitness

All of these are extremely relevant to the product development picture that this research paints. As will be discussed further in this chapter, these are key facets of the processes that are available to developers today. The process performance brought up in this model is also crucial to this work, particularly the time it takes to develop products.

In the Brown and Eisenhardt review, another key issue is brought up—that of organization of work. In their review and in subsequent work, research has shown that various processes have differing effects on development time. In particular, a highly iterative process that went through frequent reviews had a greater benefit in the computer industry than did a traditional product development process that had been compressed to reduce cycle time (Eisenhardt & Tabrizi 1995). Nevertheless, even this research had its drawbacks, specifically looking at iteration in the design process, and not necessarily throughout the development cycle. External communication is essential as well, and involves interaction between the developer and the other stakeholders, including the customer. Before looking at the processes available, it is important to understand the nature of the products involved.

2.1 Product scope

The aerospace industry is an extremely complex myriad of corporations, involving a large number of firms from material suppliers to software developers to systems integrators. The multitude of products in the industry is equally diverse. Like the corporations themselves, aerospace products include raw materials, software, manufacturing tools, electronics and avionics, satellite systems, airplanes, ground stations, and much more. While products are essentially items "sold by an enterprise to its customers" (Ulrich & Eppinger 2000), families of products are typically identified as groups of products that have common technology and are linked to similar markets (Meyer, Tertzakian & Utterback 1997). In addition to the obvious products themselves, there are a slew of related supporting products necessary for operation and sustainment of the aerospace world. Accordingly, it is unreasonable to categorize development projects by these families. Instead, it is imperative to look at metrics that quantify aspects of the product. Furthermore, due to the nature of the research in this thesis, it is possible to further pare this variety of products to slightly more complex products requiring some level of integration of modules or subsystems.

Project scope

While some products represent new categories and significant changes, others are incremental improvements on and derivatives of existing products. The difference between these is a large driver on the development process. Depending on where the product fits into the company's portfolio, and whether it is a derivative or platform, the program will pursue different strategies. The resources and time spent on each of the

various strategies will differ. To make proper decisions on resources and development processes, a program manager must be aware of the nature of the product.

Systems have a certain relationship both internally, in terms of how the components relate, and externally, in terms of their use. The terminology for such relationships is useful and relevant to the body of this work. Throughout the remainder of this work, the relationships between various subsystems within a system are referred to as the architecture of the systems. The core concepts of the system refer to the understanding of the underlying technology or subsystems. One useful method of understanding these relationships is through a framework developed by Henderson and Clark (1990) for identifying innovation types. Figure 3 depicts this framework.

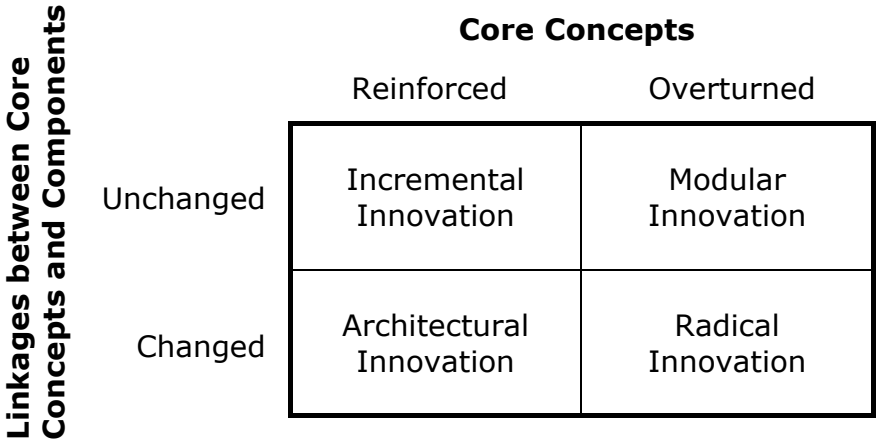


Figure 3: Henderson and Clark (1990) innovation framework

Henderson and Clark establish two pairs of extremes. The first pair is radical and incremental innovations. Radical innovation, they argue, "establishes a new dominant design," while incremental innovation "refines and extends an established design." The radical innovation changes the fundamental architecture and core concepts. The other

pair is modular and architectural innovations. The former indicates a change in the core concept, but not the architecture of the product, while the latter keeps the core the same while changing architecture.

Work by Fine (1998) describes the product development as being divided up into architectural choices, such as going to a modular or integral structure, and detailed design choices, such as performance specifications. These decisions appear to be highly relevant in implementing various development strategies. Additionally, Ulrich and Eppinger (2000) describe four major development types: new product platforms, derivatives of existing product platforms, incremental improvements to existing products, and fundamentally new products. Figure 4 shows a typical portfolio development strategy. In essence, this consists of the development of platforms and their increments, followed by derivative platforms and their increments, until the decision is made to develop a "clean-sheet" design.

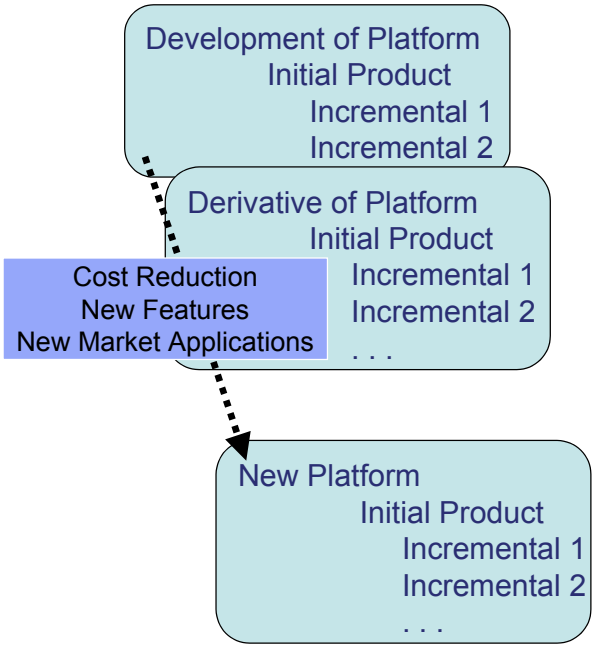


Figure 4: Ulrich and Eppinger (2000) model of family development

In other words, each of these development types is a different level investment. New product platforms tend to be a significant amount of work, as these serve as the basis for later models and derivatives. Platform examples might include commercial aircraft such

as the Boeing 777, or military aircraft such as the F-16. In these cases, the original design, or platform has served as the base of the family of aircraft. This family of aircraft is typically a mix of derivatives and incremental improvements resulting in new models. Using the example of the F-16, the numerous block upgrades represent incremental improvements in capability over the original aircraft. Derivatives of the F-16 include the Japanese FS-X / F-2 fighter. The last category represents a unique departure from the other three. Fundamentally new products represent a similar category as that of radical innovation stated above. Essentially, these products are a significant departure from the current group of products, in the way that jet engines are a radical departure from piston-driven engines.

In a similar classification scheme, Cusumano and Nobeoka (1998) describe four types of development projects:

- New design
- Concurrent technology transfer
- Sequential technology transfer
- Design modification

The new design category incorporates the platform strategy from above, while design modification addresses the derivative and incremental categories. The interesting additions are the concurrent and sequential technology transfer categories. Concurrent technology transfer refers to a transfer of information between two simultaneous or ongoing projects. This is exemplified by common cockpit design in various Airbus aircraft developed in proximity to each other. Alternatively, sequential technology

transfer looks at information passed on from prior projects into new projects, such as the establishment of standards or designs. Though numerous examples of this exist, some of the more notable examples are in aircraft avionics, where designs from prior aircraft are often used.

As a multitude of project types exist, the choice is not simple. To develop platforms is significantly more challenging, and platforms can often take two to ten times more funding and time than a derivative project (Ulrich & Eppinger, 2000). However, it occasionally becomes necessary to undertake fundamentally new products from a company perspective, for the sake of fostering an adaptive environment within the firm, and to reinvigorate the company portfolio (Ulrich & Eppinger, 2000). Recent examples are Boeing and Airbus with the Sonic Cruiser and the A380, respectively. These levels of development significance are a very important aspect of the development process, and it is hypothesized that the type of project will have a significant effect on the appropriate strategy selection.

Key development goals

In addition to these measures of product scope generally applicable across industries, there are a few development goals and tasks that have greater meaning in the military context. There are several decisions made early in the development process about what the strategic development goals of the program will be. To some degree this includes the decisions above, as to whether the product will represent a new development or a derivative of existing products, but furthermore, the decisions must be made as to whether the current development will be upgradeable in the future. In other words,

designers can choose to make the design of the product such that modifications and improvement activities would be easier to perform.

In addition to these decisions, recent work by Tondreult (2003) has demonstrated that programs often select among two categories of product development, the first based on growing performance, the second based on reducing cost. In terms of performance, this can mean two basic things. The first is a performance improvement or capability change, meaning an adaptation of a current product to new needs. The second is a design change to allow the product to comply with new regulations or changes in the technical operating environment. At the same time, developers can choose to focus on reducing acquisition and operating costs, through iteration.

Sources of inception

While many of these decisions are crucial to the development of new products, the source of demand for the new product is also critical to the development path. While the majority of these needs originate with the end user, especially in the defense community as a result of direct operational experience, other sources include Department of Defense analyses of needs, experience in wargames or exercises, and possibly new technologies or novel concepts.

Product architecture

While the scope and scale of the project represents one of the significant decisions in the development process, another more specific decision is made about the product itself. Developers often have the choice of electing to develop products with varying degrees of

modularity or integrality. The decision is important as it affects several key aspects of the product's life—it's cost, performance, and lifetime.

At the same time, the decisions made in the early stages of development will significantly affect the ability for design changes and more importantly, design innovation. In Von Hippel's (1988) work, he found that among similar products developed by different companies there was a noticeable variance in the likelihood of user-derived innovations—i.e. innovations that the user develops on their own. Further investigation showed that the cause of this was the ease of experimentation with the product. In the case that Von Hippel looked at, the products with greater innovation were more flexible, and the system could be altered more easily without the use of expensive or special parts.

Often this ability to change systems without critically hindering the system as a whole is referred to as modularity. Modularity exhibits two characteristics:

- Modules are responsible for "one or a few functional elements in their entirety"
- The connections between modules are clearly defined (Ulrich & Eppinger 2000)

The most commonly referenced example of modularity today is the personal computer (Baldwin & Clark 2000), where standardized interfaces allow for changing much of the system from the processor core to peripheral hardware. In the aerospace world, the F-16 has been referred to as an open, modular architecture (though not necessarily by the standards of other industries). The F-16 uses an avionics rack that, with some software changes, allows for different avionics modules to be put in. Conversely, some architectures are highly integrated, such as the F-22 avionics suite, which unites

information from various sources to present a single display to the pilot (Kirtley 2002). Ulrich and Eppinger (2000) describe such an integrated architecture as having some of the following aspects:

- Functional elements are distributed across multiple modules
- Single modules are responsible for multiple functionalities
- Interactions between modules are not well defined.

An example difference of the two architectures in avionics is shown below, in Figure 5.

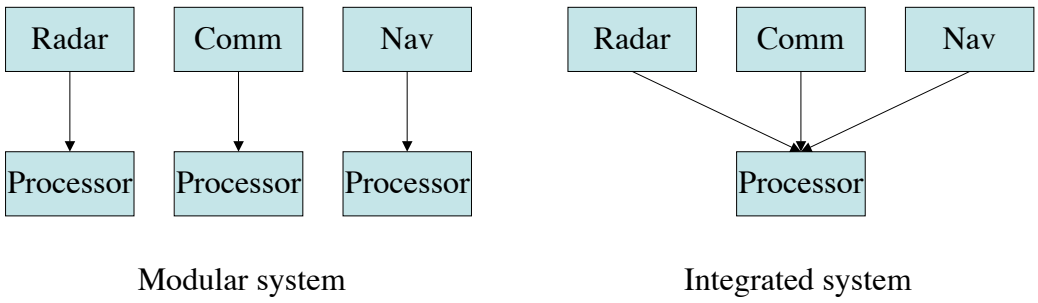


Figure 5: Modular vs. Integrated system (adapted from Kirtley 2002)

Each potentially has its own advantage. Integrated architectures, like those evidenced in the F-22 Avionics suite, have the ability to provide the user with greater ease of use, and sometimes reduce the cost of procurement. Examples of these include much of the world of electronics, where multiple chips, each performing individual tasks, have slowly been integrated into several or sometimes one chip. On the other hand, the ability to quickly upgrade these, or to incorporate changes, including new technologies or changing user preferences, is significantly diminished. Modular architectures, such as the F-16 Avionics, address this by allowing for "swapping" of parts, so that sections can be upgraded or changed, but this can create penalties in cost and weight. There are several reasons for altering products, such as (Ulrich & Eppinger 2000):

- Upgrades – the ability to change due to technological improvements or evolving user needs
- Add-ons – the ability to allow third-parties to add functionality
- Adaptation – the ability to change products based on environments and external changes
- Wear – the ability to replace individual components due to degradation
- Consumption – the ability to replace consumed materials, such as the toner cartridge in laser printers
- Flexibility in use – the ability to change products to accomplish different tasks, for example to allow for different munitions based on mission needs
- Reuse – the ability to reuse the product with only slight changes to necessary components

Fundamentally, however, the use of a modular architecture facilitates adaptability and agility. This can be critical as shown in the next chapter, since some development strategies emphasize flexibility and multiple design iterations.

Summary of products and program scope

There are a couple of key takeaways from this chapter. The first of these looks at the level of change in the product development from previous systems—that is to say how the product relates to previous products (if they exist) providing similar capabilities. This resulted in a model of family development, formed by platforms, derivatives, and increments, the first representing the highest level of development work, and the last typically representing the easier modifications to existing systems. The second major finding in the chapter was the product architecture, which indicated that systems varied in

modularity, but that modularity had significant advantages in product flexibility. This ties in to the next chapter, as the types of development (i.e. new or platform vs. incremental) as well as the modularity significantly impact the development strategy.

3 Product development processes and the role of iteration

The variety of product development processes is a natural evolution from a number of uncertainties in the lifecycle of products. Due to the dynamic nature of the market and the inherent challenges in the design and development of new products, various development strategies attempt to mitigate the uncertainties in development. This chapter specifically looks at two parts of the development process. The first of these is the type of uncertainties or risks a program can encounter during development. The second, and much more significant part of this chapter, looks at the wide range of product development processes available to program managers, categorizing them by several key attributes and the types of uncertainty they address.

3.1 Uncertainty

The uncertainties facing various product developments play a crucial role in determining what development process is appropriate. Historically, strategies have arisen as a direct result of trying to cope with development uncertainties. There are typically four types of risks and uncertainties facing new products today (Unger 2003).

- Budget - whether a successful product can be developed given the resources available, as well as whether development can successfully continue given changes in allocation of these resources
- Customer - whether the product adequately addresses the customer needs, particularly given that those needs are apt to change
- Schedule - whether the product can be developed and delivered in the time available, or in a useful timeframe

- Technical - whether the product can be built to satisfy certain requirements and performance demands.

Though these do not represent the whole of the risks in product development, they do represent a broad spectrum of the significant challenges facing firms. These uncertainties provide a good framework for which to understand the various processes available.

3.2 Process steps

Though there are many processes by which to mitigate the aforementioned risks, each of them contains some similar subset of tasks and steps essential to any product development project. While processes vary significantly in terms of the arrangement of tasks, the tasks themselves are fundamentally the same in each of the processes. They include:

0. Planning - establish a project mission statement, including targets in market and firm-level goals
1. Concept development - perform needs analysis, explore alternative concepts, and prepare specifications
2. System-level design - define architecture and subsystems
3. Detail design - full detail specifications for all parts, including manufacturing and assembly plans
4. Testing and refinement - typically consists of two major sets of tests, the first is of an 'alpha' prototype, which determines whether the product works and satisfies key performance parameters, while the second, 'beta', prototype establishes manufacturability, and is often tested in a real-world environment

5. Delivery - full-scale manufacturing of the product, accompanied by the launch or delivery of the product (Ulrich & Eppinger 2000).

These steps are the building blocks of any development process and form the basis for discussion of the various processes. A visual representation of this is often shown in the Vee model, a popular visualization method for project managers. Despite the different naming of the steps that place it in a software context, the process is essentially the same.

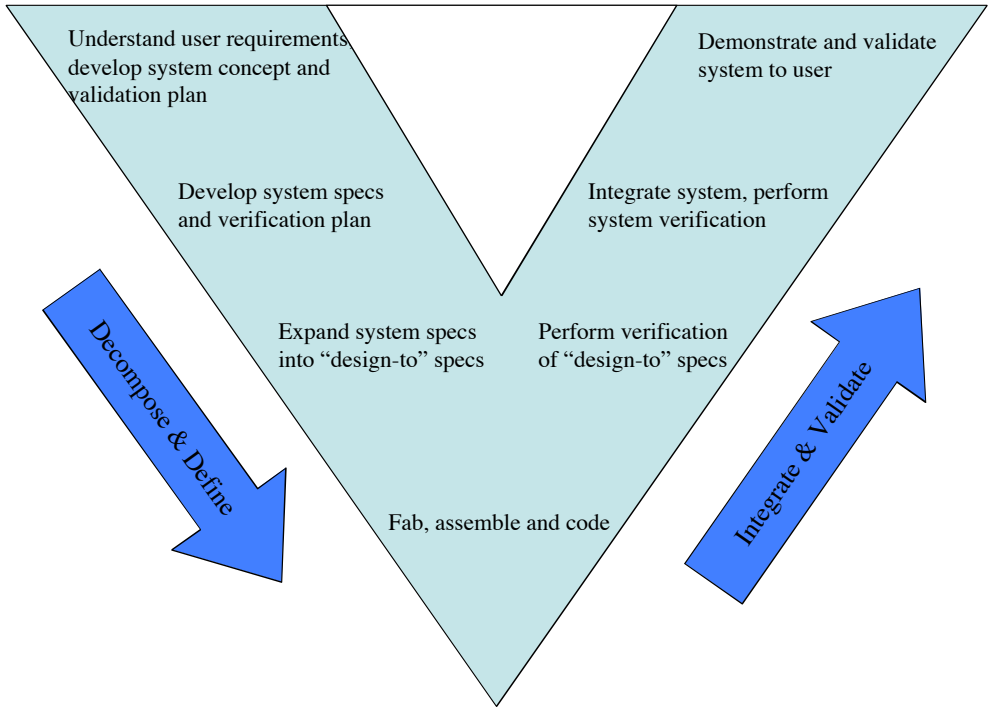


Figure 6: Vee model (Forsberg, Mooz & Cotterman 1996)

Note that in Figure 6, essentially the same set of steps is required, but there is more detail on the testing and validation portion of the development process. This has been a significant part of the thrust of the Air Force's improvement program, as testing proves to be a significant challenge from both a schedule perspective as well as the role which testing should play. The testing steps include verifying "design-to" specs and performing system verification, which run along the lines of the Air Force's Developmental Test and

Evaluation, while demonstrating and validating the system is akin to Operational Test and Evaluation (for more on this, see the testing section below).

3.3 Key program tasks

In addition to the general tasks performed in any development, there are a number of key activities that must be performed. These typically are done as part of the general task categories, but are crucial to the development process and deserve additional discussion. In particular, these tasks are identified as being crucial to spiral development and other highly iterative processes.

Stakeholder identification

One of the most important parts of any product development task is identifying the key stakeholders. This is often done through a lifecycle analysis, where communities that play a role in the product's lifecycle are identified. These do not make up the entire stakeholder community, as stakeholders are often tiered, with major stakeholders often having their own groups of stakeholders. Though stakeholder significance varies, within a typical military development program, these stakeholders consist of the end user, logistics, the acquisition customer, the contractor, and testers. Through a dialogue between representatives from each community, a roadmap for product development is arrived at.

Modeling and simulation

Modeling and simulation represent a critical aspect of the development process. They are often used in hardware development processes as a preliminary means of understanding the product's performance. In particular, models and simulation can provide a surrogate for physical production, the latter often being very costly, and much more difficult to

change. At the same time, these can be very valuable system representations for soliciting feedback from the various stakeholders, including the user and customer (Dare 2003). Through this feedback process, the program has the option to evolve and / or verify requirements. Due to the relative ease of change, models and simulations can be modified to incorporate new technologies, as well as new requirements for future systems. This is particularly valuable for programs intending to have many increments. The Department of Defense (DSMC 2001) defines a model as "a representation of an actual or conceptual system" usually based on mathematical algorithms, while a simulation is essentially the use of these models to determine usability of these products.

Prototyping

Prototypes represent a more evolved system, and are traditionally at a level of definition higher than models and simulations. They are often "an original or model on which a later system / item is formed or based" (DSMC 2001). The term applies to both software and hardware systems, though software systems have slightly different nomenclature, for example alpha and beta versions and so forth. Prototypes can represent either a part of or a whole system. In the case of hardware, prototypes usually refer to physical representations of the product. In the case of aircraft, for example, this might include prototypes of the cockpit, and later full-scale functional prototypes. The end purpose of prototypes, like simulations and models, is to provide an opportunity for feedback from the stakeholders and to modify and validate requirements. As prototypes become more complex, their flexibility is often reduced, so it becomes imperative to feedback-critical programs to deliver prototypes early. The role of prototyping varies greatly among the various product development processes, ranging from a means of requirement solicitation

in spiral development, to a means of final verification and validation in traditional waterfall processes.

Testing

Testing is a fundamental part of the development process, and crucial to aerospace programs, where error is intolerable. The Department of Defense defines three types of testing, two of which are relevant to this research. The first of these is developmental testing, defined as

1. *Any testing used to assist in the development and maturation of products, product elements, or manufacturing or support processes*
2. *Any engineering-type test used to verify status of technical progress, verify that design risks are minimized, substantiate achievement of contract technical performance, and certify readiness for initial operational testing* (Defense Systems Management College 2001)

Essentially, this is the technical certification testing, ensuring that the product performs reliably and safely, and two the requirements determined by the developer and acquisition community. The second level of testing is operational testing, described as:

1. *The field test, under realistic conditions, of any item (or key component) of weapons, equipment, or munitions for the purpose of determining the effectiveness and suitability of the weapons, equipment, or munitions for use in combat by typical military users; and the evaluation of the results of such tests* (Defense Systems Management College 2001)

This testing serves to identify the usability of the developed product. Both types of testing are developed in the Test and Evaluation Master Plan (TEMP). The TEMP

defines the structure and goals of the test program. Recently, the defense acquisition community has been pushing for an integrated or combined testing and evaluation, involving both developmental and operational testing. This will hopefully reduce some of the redundancy and reduce acquisition time.

Technology insertion

Though the reader is likely very familiar with the definition of technology, the context of these technologies is hypothesized to play a significant role in the development strategy. This focuses on two aspects, the first being the maturity of the technology, and the second being the rate of change of technology. Technology maturity has long been known to reduce cycle time, as programs can quickly move to insert this technology without significant changes later in the development. In other words, the more mature a technology, the less likely it is to change, and the easier it is to incorporate into a development process. Though there are a number of categorization schemes for identifying the maturity of products, most notably the Technology Readiness Levels (TRL) determined by the Department of Defense, they are somewhat cumbersome. More simply, off-the-shelf (sometimes referred to as Commercial-Off-The-Shelf, or COTS) technologies are regarded as relatively mature. In addition to this, companies, and especially the military, often have previously developed technologies, usually called Non-Developmental Items (NDI), also seen as mature. Beyond the level of maturity, companies must consider the evolution of technology in their design decisions. Technologies often evolve much more rapidly than the products in which they are a part, leading to a various outcomes. In selecting product architecture, the ability to incorporate new technology or exchange older parts for newer ones becomes essential, allowing the

program to be flexible and adaptive. The inability to do so can not only mean a shorter lifespan for the product, but also that maintenance of the product can become a problem as a problem of diminishing manufacturing sources arises. As a result, the product development process must negotiate a path between mature components and components that can evolve in order to provide the best lifecycle value.

Summary of program tasks

In every product development process, these activities must be successfully negotiated. At the same time, each task has different priorities in each of the strategies. More importantly, each of these tasks holds different weight depending on the development process. To review, these consist of the following: planning, concept development, system-level design, detail design, testing and refinement, and delivery.

Using these steps and the uncertainties from above, it is possible to evaluate the portfolio of product development strategies. These will be broadly defined in their families, and further attention will be paid to the detailed understanding of individual processes. Finally, an overview of the Evolutionary Acquisition policy and current military development processes will be provided. Various arrangements of these product steps yield approximately seven strategies in four major categories. These include waterfall processes, evolutionary processes, incremental processes, and spiral development. Each of these will be illustrated in more detail in the sections below.

3.4 Waterfall Processes

The waterfall process is easily the most recognizable process available in the project manager's toolbox. It has long been the standard in the industry, undoubtedly due to its

intuitiveness and straightforwardness. It is, very simply, the sequential and linear organization of the aforementioned product development steps. Figure 7 shows the waterfall process.

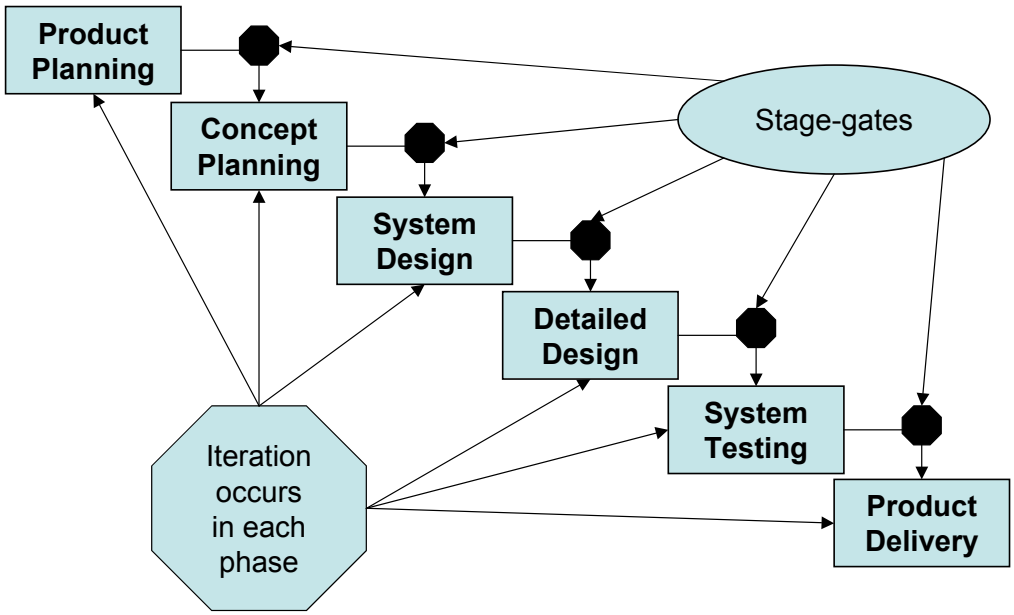


Figure 7: Waterfall process (adapted from Unger 2003)

In the waterfall process, steps are intended to be executed only once, and accordingly, there is an accompanying set of stage-gates. These stage-gates are essentially milestones or reviews, serving to verify that the stage has been successfully concluded, and work can move to the next stage. The result is a process that has a high level of iteration within individual stages (e.g. the system design stage may be repeated several times), but not across stages. In the waterfall, it is presumed that once the product has passed a certain phase, there is no need to return to that phase. When changes are required in earlier work, it comes in the form of undesirable rework. The result is a single product that should provide a best match to the up-front requirements or determined market need.

The Apollo program is one such program that used a waterfall process. This was particularly valuable to the program as it was faced with a technical challenge but the design goal was clear. The waterfall process handles technical risk very well, as each stage serves to ensure that the technical aspects of the program have passed inspection and are complete.

There are some notable advantages to a waterfall process—primarily the structure it provides to the project. Requirements and specifications are frozen early on, which can be advantageous as it reduces the likelihood of changes further along in the process, when they typically cost more. Additionally, because of the rigid nature of the process, budgets, schedules and resources can be somewhat more predictable. In fact, the effort up-front in planning can reduce development time by eliminating unnecessary steps and improving coordination and communication (Eisenhardt & Tabrizi 1995). Tools such as Design Structure Matrix (DSM) are commonly used in optimizing these processes. Additionally, it is accepted that in cases where the requirements are well defined, it is risky not to use the waterfall process (Boehm 2000, Highsmith 2000).

At the same time, the waterfall is unresponsive to dynamic environments, such as budget cuts, new technology insertion, and changing user needs. In fact, the cost of change is estimated to go up an order of magnitude with each stage (Owens & Cooper 2001). Additionally, the method often lends itself to an 'over-the-wall' approach, where parties responsible for each stage pass the work over to the next group with little interaction.

This can cause delays and cost increases, as well as the potential for poor quality products. Waterfall processes do not provide interim capabilities, and requirements are not flexible in the process. While the process is initially driven by user requirements, there is no real opportunity for user feedback except the occasional milestone review, but changes at these stages typically require rework. In addition to this there is no intentional iteration across stages, as stages are intended to be frozen once executed.

Waterfall variants

While the waterfall represents the traditional process of choice, some of the weaknesses of the process have driven project managers to develop variations on the strategy.

Parallel waterfalls

Due to the nature of the waterfall it is impossible to proceed to a subsequent stage without completing the prior stage first. This can cause problems if and when stages are held up. Particularly, many aspects of the system and detailed design can encounter technical challenges. Accordingly, the parallel waterfall model mitigates the risks of overall delays by breaking the design task up and executing these parts simultaneously, as shown in Figure 8.

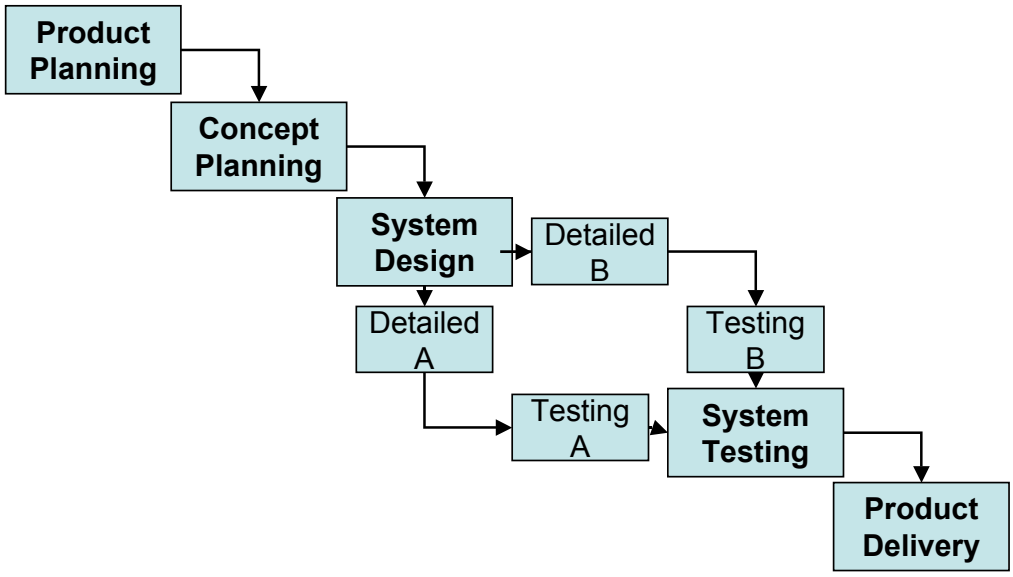


Figure 8: Parallel waterfall process

The figure shows one example of a parallel waterfall, where the detailed design paths take different routes. The parallel waterfall starts like a normal waterfall, but allows for a variety of designs to be considered within the context of the system design. In such a program, a set of early design parameters is necessary to facilitate individual work on the separate detailed design sections. Typically these sections will consist of subsystems that are farmed out to subcontractors or other divisions. Furthermore, the parallel waterfall system can be used to develop a portfolio of design options, such that if one strategy fails there remain alternatives.

A common aerospace example of a development that would use a parallel strategy is the co-development of an airframe and engine. In other words, to build a new aircraft with given performance, the program sets the requirements and allows for simultaneous development of the airframe and engine so as to reduce overall development time. Again each of the developments is like a waterfall in itself, so the process is still primarily

technical risk oriented, but parallel processes allow for some mitigation of schedule risks to the overall program due to one subsystem.

The parallel waterfall essentially has two key benefits over the traditional waterfall—first, it allows for work to be completed in parallel, so that delays in one path do not impact the final schedule as significantly, and secondly, to provide a portfolio of options in cases where customer needs are uncertain. At the same time, there is a significant requirement that the strategy places on the design. The design must be such that it can successfully be subdivided without coupling between the various divisions. If the system is highly coupled, then a change discovered in one path could potentially require rework in all paths. In this regard, there must be a very clear set of interface requirements, and the system must exhibit some level of modularity. Though the parallel waterfall process mimics the traditional process, it does allow some flexibility in that it can incorporate multiple simultaneous designs.

Compressed or overlapping waterfalls

Another variant of the waterfall is the 'compressed' or 'overlapping' waterfall. While the traditional waterfall lends itself to an 'over-the-wall' approach, the overlapping waterfall attempts to mitigate this problem by creating an overlap, whereby functional teams interact for some period before the subsequent team takes over, as demonstrated in Figure 9.

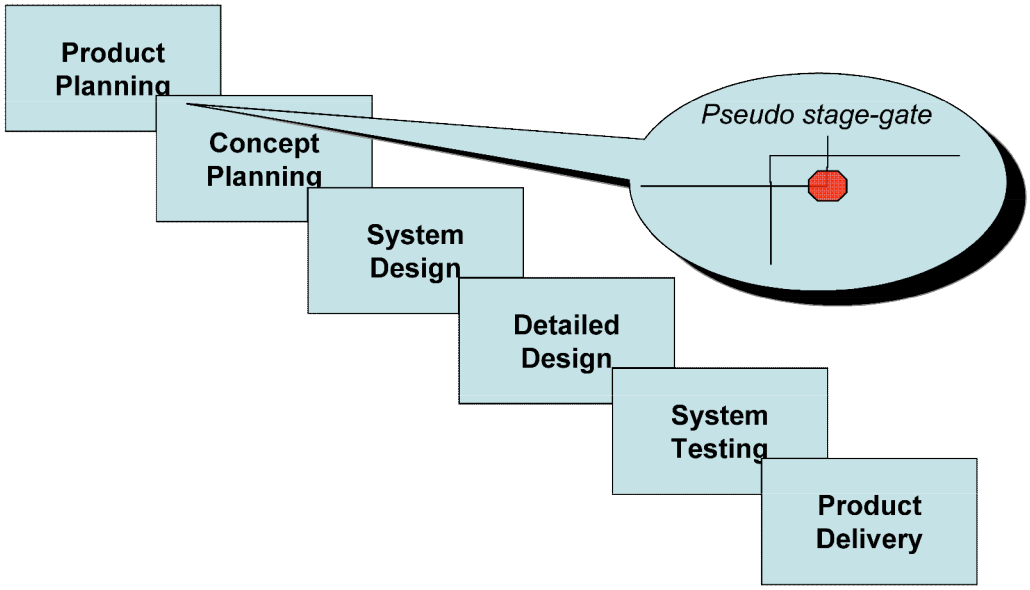


Figure 9: Overlapping waterfall process

Note that the stage-gates in the overlapping waterfall are not at rigid as in the traditional process. An additional advantage to this process is that it allows some parts of each phase to be done in parallel, compressing the development time. With overlapping stages cross-functional teams are more likely to occur, which is generally considered to be beneficial to innovation. Furthermore, involving teams earlier in the process and alongside the prior team decreases the likelihood of rework later as well as reducing the wait time between steps. This also increases the project orientation of the program, rather than the function orientation, which is likely to benefit the enterprise (Unger 2003; Cusumano & Nobeoka 1998).

This development strategy is very close to the pure waterfall, and it is likely that programs suffering from challenges in cross-functional communication might tend towards such a process. At the same time, the lack of clearly defined reviews or stage-gates can be a drawback of the overlapping process. Additionally, though it offers some

shortening of cycle time in comparison to traditional processes, the mere compression of phases has not been shown to have dramatic schedule benefits (Eisenhardt and Tabrizi 1995). Like the waterfall process, there is little or no real flexibility or iteration in this process, and accordingly no opportunities for interim capabilities or user feedback.

Design to schedule / budget

One of the largest drawbacks of the waterfall and its siblings is the lengthy schedules and corresponding costs. These are especially apparent when programs face schedule slips and cost overruns. In the design to schedule or budget strategy, the program is aggressively fixed to a target schedule or cost, at which point the program will stop that work and move to the next stage. Figure 10 shows this graphically.

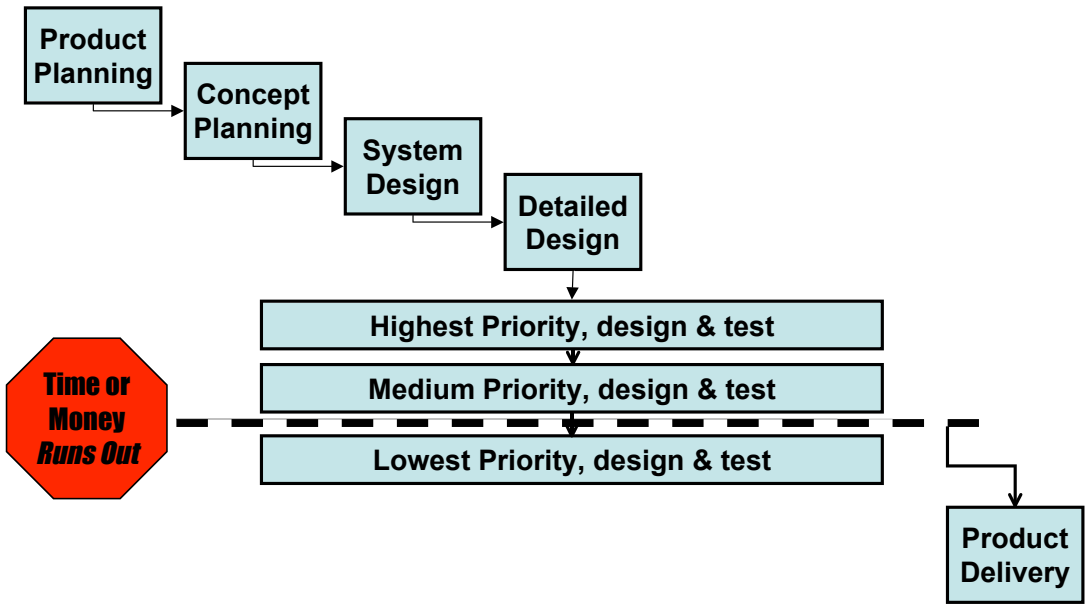


Figure 10: Design to schedule / budget (adapted from Unger 2003)

Note that in Figure 10, the first phases are the same as the waterfall, but the detailed design are prioritized and subsequently developed in that order. At some point a schedule or budget target is reached, and the program moves to the following phase, which is the final delivery phase (Unger 2003).

The significant advantage to this process is the clear cost and time targets, which effectively prevent overruns encountered in traditional waterfall processes from occurring. The Air Force has begun to use this methodology in a Cost-As-an-Independent Variable (CAIV) or more recently a Schedule-As-an-Independent-Variable (SAIV) strategy. These effectively set performance parameters based on what is considered feasible given a cost or timeframe. There are significant advantages to this, and graphics chip maker Nvidia used such a strategy to capture a large market share by rolling out new products every six months (Turner 2002). At the same time the process requires careful management—improperly prioritized tasks can lead to a poor product. Additionally due to the level of iteration within the design phase, and the uncertainty of what tasks will be completed, interdependent tasks are undesirable and require special planning to work, as incomplete tasks cannot prevent product functionality.

Most any development process uses a schedule and budget target, but rather than execute on a larger set of requirements and complete only those possible given those targets, typical programs select requirements up-front based on what is considered possible given the budget and schedule. As the process resembles a waterfall, it has no inherent flexibility or useful increments. Alternatively however, such a process can be used as

part of an evolutionary strategy, with individual iterations using a design to schedule or budget process.

Summary of waterfall processes

The waterfall process is a very powerful tool in the program manager's portfolio in dealing with specific risks. It works particularly well when the program requirements are well defined and static. While the traditional waterfall handles technical risks best, its variants also attempt to mitigate other uncertainties, such as schedule and budget.

3.5 Evolutionary processes

A significant drawback to the waterfall and its variants discussed so far is their inability to incorporate feedback late in the process, which is of special concern in cases where requirements are uncertain to begin with. Evolutionary processes, which should not be confused with Evolutionary Acquisition (despite some similarities), attempt to address some of these issues. In particular, they attempt to reduce the likelihood of quality and user issues through the development of functional products that can be used and tested, i.e. prototypes or deliverables, as shown in Figure 11.

This process goes by two names, evolutionary prototyping and evolutionary delivery. Both processes essentially go through a waterfall-like development until a functional prototype is developed.

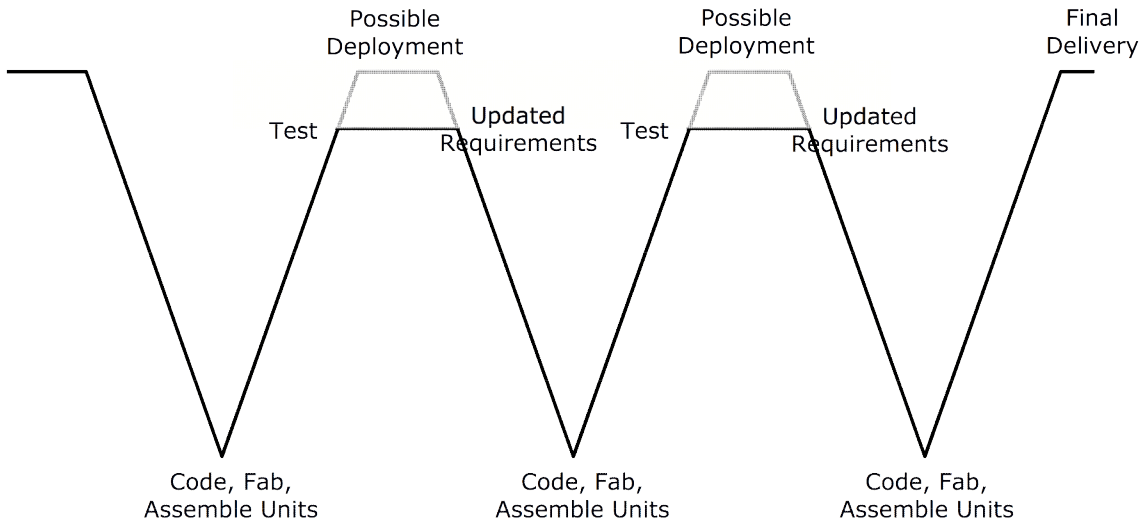


Figure 11: Evolutionary prototyping and delivery (adapted from Forsberg, Mooz & Cotterman 1996)

The use of evolutionary development can be very beneficial to the development process in that it gives stakeholders a tangible item to be worked with. It is also useful for applications where the user has some idea of what the end product should be, but not a complete vision—the oft cited "I'll know it when I see it" mentality. It can also be valuable in understanding real-world applications of theoretical models. Fundamentally, the system focuses on prototypes and continues to develop and iterate these until the stakeholders are satisfied.

Software programs often use this strategy to quickly deliver a tangible system to the customer and iterate rapidly. The program must have a general direction and some set of requirements to use this strategy, or it runs the risk of wasted effort. Typically these processes are used to finalize unknown requirements, not to generate them. In other

words, some part of the design is already known, and the program iterates on the remainder.

It requires, like many of the waterfall variants, several program abilities to be a successful process. First of all, the program must have the ability to rapidly prototype (at a reasonable cost). Secondly, the program must use an architecture such that design changes can be rapidly performed. While the process does deliver more useful products to the end user, it can potentially lengthen schedule and increase costs due to the time and money necessary for prototyping. Additionally, the process can also introduce some of the stability of a waterfall process through prototyping, preventing significant or excessive changes in the core design, and allowing only modifications. Evolutionary processes differ in this way from waterfall programs, as there is an opportunity for user feedback. Though it is possible to provide the user with interim capabilities, this is often not the case, as prototypes or near-final deliverables are usually few in number.

3.6 Incremental processes

In response to the longer cycle times typical to waterfall processes, especially those due to technical uncertainties, an alternative process is an incremental delivery process. Forsberg, Mooz, and Cotterman (1996) propose two types of incremental developments. The first of these is an incremental development with a single delivery, while the second is delivers multiple increments. The latter is part of the Air Force's Evolutionary Acquisition strategy, and will be discussed in further detail later on.

Incremental development with a single delivery works well when a single delivery is desired, but requirements are well known (Forsberg, Mooz & Cotterman 1996). Once again, this benefits from some level of modularity, as the system must be able to function as a single whole despite being delivered in increments. The authors site the example of the Space Shuttle, where main engines were awarded in 1970, the orbiter in 1972, integration in 1980, and first flight in 1981. For the purposes of this research, this is not a focal point of development processes.

Incremental development with incremental deliveries, however, has been a strategic focus of the Air Force for some time, and despite a several names for this, this has been the preferred strategy for large acquisition programs. The process delivers increasing capability over several (or more) stages (Forsberg, Mooz & Cotterman 1996). Figure 12 depicts an incremental delivery using the Vee model.

As is shown in Figure 12, the plan for incremental deliveries is determined up front. In other words, the plans for the later deliveries are primarily determined in the early requirements process.

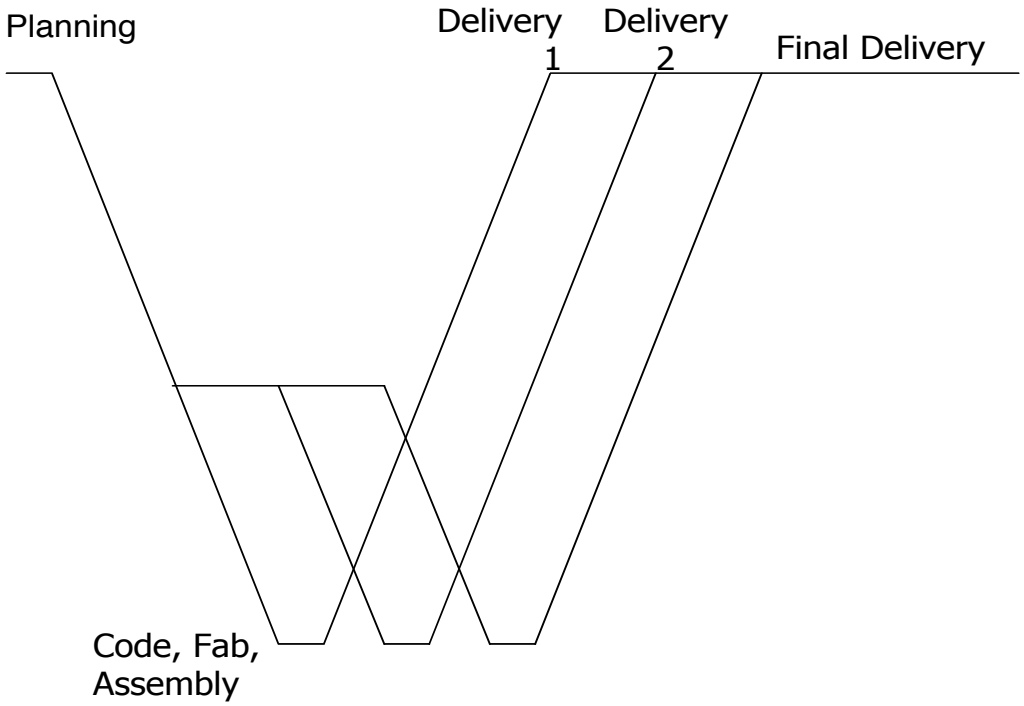


Figure 12: Incremental delivery (adapted from Forsberg, Mooz & Cotterman 1996)

The incremental process offers a significant advantage over waterfall processes in offering earlier delivery, as advanced technologies can be placed in later deliveries, so that there is not a wait time for technology maturation. In addition, the process offers some ability for learning between increments, but again due to the up-front development work, significant changes are more challenging.

Incremental processes typically are used on a number of programs where particular development aspects of the product are not expected to complete at the same time. Nevertheless, a basic capability can be delivered if the program does not wait for the additional sections to be completed. The Predator program had several planned improvements during the course of its development, though the main aircraft was

delivered well before these later systems. A more simplistic example is train tracks. A new rail development might plan for a length of thirty miles, but deliver these in ten mile increments—useful, but not the full capability of the system (Forsberg, Mooz & Cotterman 1996). Nevertheless, the first ten miles serves as the basis for the additional increments. Programs might do this for financial reasons as well. If funding for development of a given subsystem does not initially exist, a program might choose to field what it can and deliver additional capabilities as possible. Incremental processes begin like waterfall processes, as planning takes place typically for multiple increments up front. Nevertheless, as increments are fielded, there is an opportunity to make changes to increments underway or as yet not begun. Accordingly, the user has an opportunity to influence the final outcome of the product. Additionally, due to the nature of the process, there are usable interim products and capabilities.

Military processes

The military uses two additional, but similar processes, focusing on incremental deliveries. The first of these is Pre-Planned Product Improvement (P3I), and the second is a Block upgrade program.

P3I attempts to accommodate for change by allowing for some flexibility in the design for future upgrades and changes. In particular, it is used to allow for technology improvements over current capabilities, or in cases where initial budget and schedule do not allow for certain capabilities. The added flexibility allows for evolving needs, and possible extension of the product operational life. It is also benefits the user by speeding up delivery, as the program does not wait for technology maturation (DAU 2000). At the

same time, it requires better long-term planning, and can increase up-front costs. These programs are typically defined upfront with staged execution.

Block upgrades are also incremental deliveries, and can often incorporate the P3I capabilities in an upgrade. Block upgrades are not completely planned for up front, and can be planned for either at the outset of a program, during a current development, or occasionally after development work is done (though this is rare). However, the block upgrade typically occurs after delivery of the previous block, and incorporates user feedback and dynamic needs. Figure 13 shows one representation of the Block upgrade methodology.

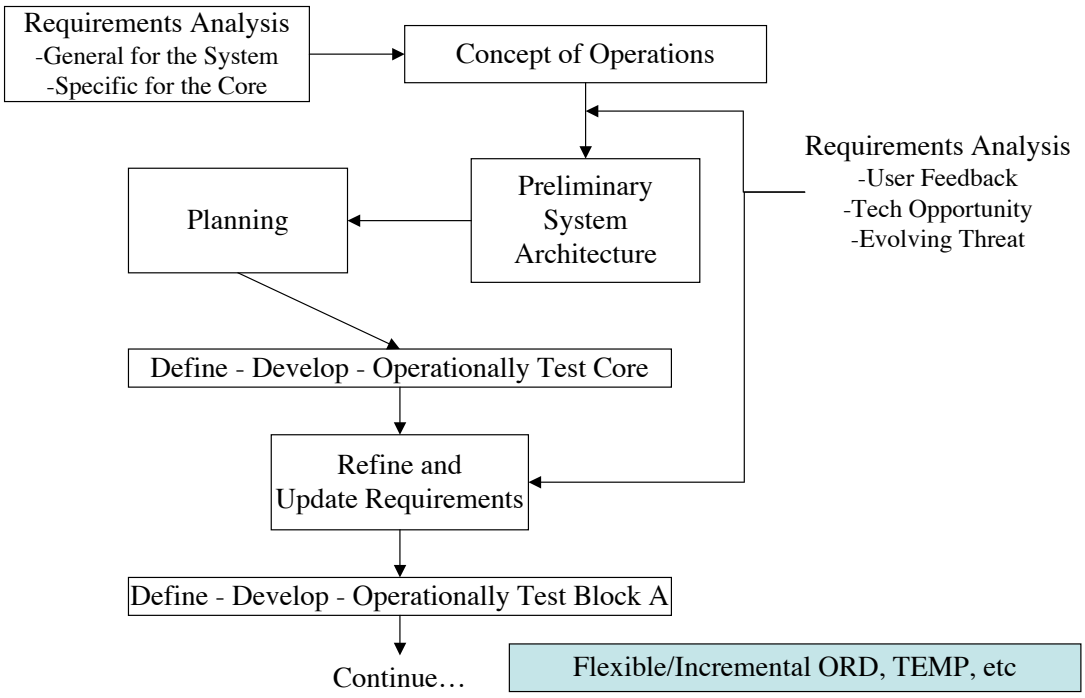


Figure 13: Block methodology (adapted from Defense Acquisition University 2000)

Note that the figure shows the milestones used by the Department of Defense, but essentially the development process occurs similarly, with a full waterfall to deliver the first block, and subsequent developments usually appearing as truncated waterfalls, often

starting in the detailed design or system design phases. The Block upgrade typically relies on full capability in the first delivery with Block upgrades improving this capability over time. Essentially this has been the primary strategy in the Air Force for incorporating new capabilities.

3.7 Spiral development

Spiral development has become a strategic focus of the Air Force acquisition strategy in the last several years. In fact, while evolutionary acquisition is the desired methodology for acquisition, spiral development is the preferred process (Wolfowitz 2002). At the same time, spiral development, in part because of its origins, has been a challenge to apply to Air Force systems.

The origins of the spiral

The spiral process as it is perceived today appears to have come about in the 1970's as a direct result of the new era of heavy software involvement in technical products. In particular, as products depended more heavily on software for functionality, the challenges facing software development became more apparent and more significant. Software quickly became an entity in its own right, and further challenges arose, especially as people were uncertain as to what they wanted. Detailed up-front requirements and specifications were not as valuable in this type of uncertainty, and a focus on tangible prototypes to solicit user feedback was very effective. Furthermore, software became and continues to this day to be the first resort for adapting to late requirement changes. As a result, the need for a flexible development process was imperative. More recently, with Evolutionary Acquisition and spiral development, the Air Force is pushing to take spiral development out of the software world and apply it to

hardware as well. This provides additional difficulties in both implementation and understanding. The remainder of this section will attempt to clearly explain the process and the various interpretations, and provide a clear understanding of what spiral development means for systems acquisition.

Spiral development and iteration

Like the processes before it, spiral development is a risk-mitigation process, but at the same time, it is distinct from these processes as well. According to Boehm (2000), the strategic goal of the process is "to guide multi-stakeholder concurrent engineering of software intensive systems." This goal of stakeholder involvement is one aspect of spiral development that is important, but certainly not unique. What is makes the spiral process special are its two main features:

> *One is a cyclic approach for incrementally growing a system's degree of definition and implementation while decreasing its degree of risk. The other is a set of anchor point milestones for ensuring stakeholder commitment to feasible and mutually satisfactory system solutions* (Boehm 2000).

The cyclic approach Boehm refers to is simply an iterative process across the whole of the product development process (sans delivery). In other words, the program should begin with concept development and continue through to some level of design (it could be system or detailed depending on the project). This is followed by developing a system representation, often a prototype, using this as a means of soliciting feedback from the user and other stakeholders (testers, logistics, etc.).

The Boehm spiral is shown in Figure 14, and consists of a spiral and four quadrants with four overarching steps, involving determining and evaluating risks and constraints, developing (and verifying) the next iteration of the product, and planning for future phases.

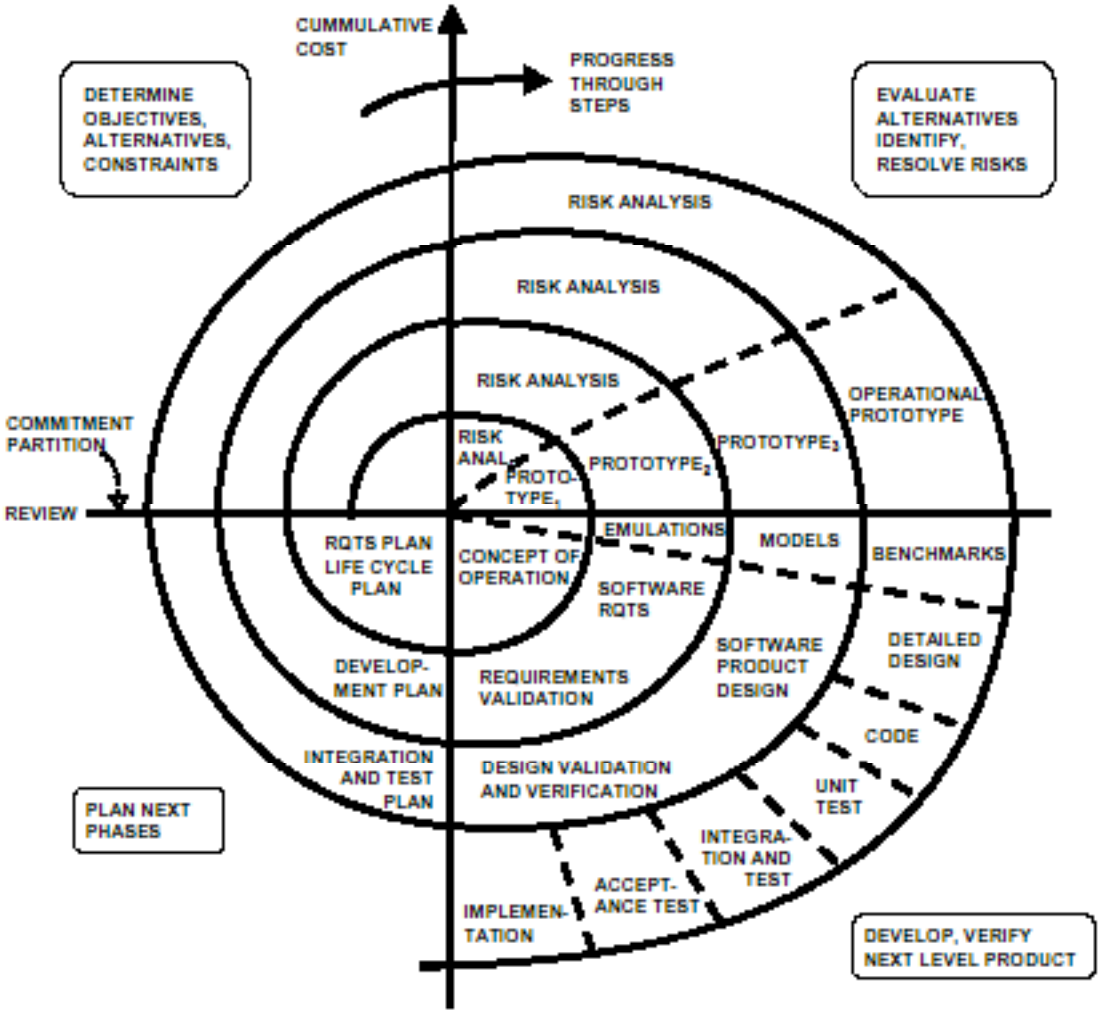


Figure 14: Boehm spiral model (from Boehm 2000)

As shown in Figure 14, the process is somewhat complex, and difficult to understand. Fundamentally, spiral development asks two questions:

1. *What should the project do next?*
2. *How long should the project continue doing it?* (Boehm 2000b)

The traditional spiral also is broken up into sub-steps. The first is determining objectives and constraints, where the developer should identify the entire life-cycle stakeholders, establish initial system boundaries and external interfaces also identifying situations in which there might be negative outcomes for the stakeholders. Secondly, the developer must identify and evaluate alternatives, gathering essential requirements and options from the stakeholders, and evaluating them against the potential positive and negative outcomes. At this stage, before development, the risks must be identified, planned for, and managed. Next, the program proceeds to development, with the understanding of what areas of the product and life cycle need to remain flexible. Finally, completing the spiral involves further elaboration on each of these steps, planning for the next spiral.

Boehm also identifies three key milestones: Life Cycle Objectives (LCO), Life Cycle Architecture (LCA), and Initial Operational Capability (IOC). The first of these is to ensure that one of the alternatives makes sense from an enterprise perspective (i.e. is feasible and addresses the needs). The second, LCA, is to determine the single option to pursue, and finally, the IOC is the first operational product. At each of these milestones, the project team must review six critical issues: operational concept description, prototyping results, requirements description, architecture description, life cycle plan, and feasibility rationale.

Another significant issue is the collaboration of multiple stakeholders in the development strategy. The involvement of stakeholders early in the process is crucial to the development path. In essence, early input by the various communities, including the end

user, acquisition community, engineering, manufacturing, testing, logistics, and so on, will allow the program to ensure that key development decisions do not cause problems subsequently in the development process. This is very much the same lesson learned in the aerospace industry about design for manufacturability, and creating Integrated Product Teams (IPTs). Moreover, spiral development is a strategy to specifically involve the user or customer in the development process as part of the decision maker throughout the program. Furthermore, spiral development provides an organization the opportunity to keep the customer in the loop, and allow them to keep the program progress in line with the user needs. Succinctly put, the spiral is

> *An iterative process…[it] provides the opportunity for interaction between the user, tester, and developer. In this process, the requirements are refined through experimentation and risk management, there is continuous feedback, and the user is provided the best possible capability*. (Wolfowitz 2002)

Spiral Implementation

The complexity of spiral development also implies several key aspects about its implementation. In particular, spiral development requires a number of key elements to be pursued in both the development process and the product.

From a product perspective, the first release must be scalable enough to adequately address the user needs over the product lifecycle (Roberts 2003), such as performance, safety, and so on. To some degree this implies modularity, or at very least the ability to improve or grow product performance until some end state is reached. Though this might

be possible given an integrated architecture, the challenge in making design changes would prove to be very difficult, and time and resource intensive. At the same time, the product must be able to evolve to replace existing legacy systems. This requires careful selection of product architecture to both succeed current systems and to evolve to future capabilities.

From a process perspective, developers must keep in mind several things while performing spirals and iterations. First and foremost, the initial release must be at a state where it is meaningful to the stakeholders, i.e. that the stakeholders can provide valuable feedback for successive iterations. Secondly, it must satisfy the expectations of key system stakeholders. This in itself means two things. For the developer, it means understanding the other stakeholders' needs in such a way that the first release or prototype is along the lines of what they need. For the other stakeholders, it is the realization that the first release or prototype is the first in a series and does not represent the final state. At the same time, the operational user organizations must be flexible enough to "adapt to the pace of system evolution" (Boehm 2000b). All of this involves a significant level of stakeholder involvement, and the ability of the developer to rapidly interact with the other key stakeholders to make decisions.

Spiral development processes have the potential to offer shorter times between program inception and initial operational capabilities, but not necessarily. This requires the use of mature technologies--otherwise, the program is essentially maturing new technologies, and this takes time. At the same time, the iterative and experimental nature of spiral

development offers opportunity to involve newly ready technologies and quickly insert these, if an appropriate architecture is chosen. Both the spiral and incremental development strategies offer much shorter cycle times in delivering capabilities. The key difference is the flexibility in requirements, and heavy evaluation of interim products by the key stakeholders in spiral development. One common misconception of spiral development is that it reduces cost and schedule to final capability. This is not necessarily the case. The planning of other strategies often focuses on cost and schedule and accordingly reduces cost and schedule. Certainly the spiral is not the quickest means of getting from point A to point B, when compared to traditional waterfall processes, but it ensures that upon arriving at a final (or next) capability, it satisfies the stakeholder needs. This is essentially the strength of the process according to its proponents—making sure the product satisfies user needs.

Process resources

At the same time, because of the inherent flexibility of spiral development, there are greater burdens placed on management. Spiral processes are often very management intensive, requiring particular attention to resource allocations within a dynamic program (Highsmith 2000). Within a given spiral, assessment of resources will often be more accurate, due to the shorter cycle times. One of the larger problems experienced with hardware spiral acquisitions is in logistics. Because of the rapid nature of spirals, the potential for multiple versions exist, causing logistical issues with maintenance and deployment. In addition to this it is essential that programs are fully funded (DAU 2000), meaning that funding should be stable and accordingly, budgeting staff should be heavily involved in the program. Nevertheless, little research has been done on the ability to

successfully implement spiral development in the face of budgeting challenges. Theoretically, one would anticipate that the flexibility of the spiral process would allow for flexibility despite budget cuts, on the other hand, the ability of spiral development to generate new system representations is essential, and the program's ability to retarget to a new goal is not necessarily cost effective. Accordingly, it is a subject matter for further research. Furthermore, requirements management is a significant concern especially as requirements are not completely defined in the spiral process. There is no doubt that resource management will be different for various processes, and especially spiral development.

Spiral interpretations

As a result of the complexities of spiral development, there have been a number of interpretations of what spirals should entail. The most obvious of these is the Evolutionary Acquisition with spiral development strategy put forth by the Defense acquisition community. This focuses on delivering useful capability on each spiral and providing for use and evaluation between cycles, shown notionally in Figure 15.

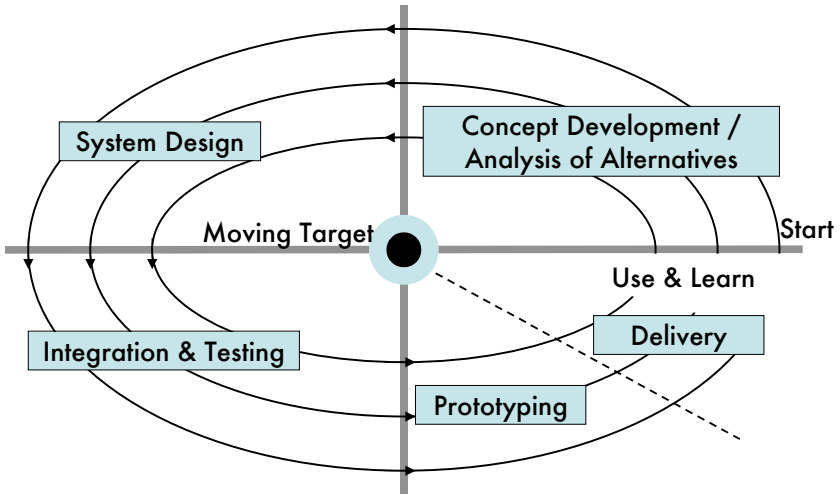


Figure 15: Notional Defense acquisition spiral process

The figure indicates several aspects of the spiral process. Unlike the Boehm model, spirals become smaller with each iteration. This is to emphasize the target-like nature of the spiral. In other words, each spiral brings the product closer to an end target, or final operational capability. Specifically, the Department of Defense (2003) defines spiral development as:

> *Desired capability is identified, but end-state requirements are not known at Program Initiation. Requirements for future increments dependent upon technology maturation and user feedback from initial increments.*

Fundamentally, this reiterates several of the key aspects of the Boehm spiral, but rather than placing an emphasis on system representations, the Defense strategy mandates capabilities, or useful increments.

An alternative representation of this model is a three dimensional spiral, with time on the vertical axis (coming out of the page in Figure 15), and allowing for various decisions to be made at each spiral, such as the decision to continue with the program, if a suitable capability is achieved or if the program is not achieving acceptable performance. If the choice to continue is made, there is a second choice of producing the results of the spiral, or as the case may be just modeling or prototyping the performance in that spiral. This alternative allows for some alleviation of the logistics issues associated with the current military interpretation of spiral development.

Summary of spiral development

Though there are a number of interpretations of spiral development, there are common aspects to each of these, providing a useful framework for understanding the process.

First and foremost, the spiral process is a highly iterative process, using some deliverable to solicit greater interaction between various stakeholders and reducing risk, while at the same time, the increasing product certainty, including requirements. At the same time, there is an underlying assumption that the product is scalable, and can be grown to satisfy stakeholder needs. This implies a level of modularity and technology maturity, such that the focus of the development process is on delivering useful capabilities to the acquisition community. At the same time, spiral development requires significant commitments from these stakeholders, necessitating heavy involvement by all major decision makers and players in the product lifecycle. Fundamentally, the spiral process is focused on mitigating various sources of risk through cautious iteration, rather than heavy up-front planning. Essentially, this means that the process is oriented towards maximum flexibility in requirements and slowly stabilizes these requirements until the end goal is reached. It can do this by supplying useful interim capabilities. The basic nature of the spiral is also intended to create a maximum level of iteration across the whole of product development steps, so as to delay the program from limiting product options.

3.8 Process summary

Despite the variety of processes, there are a number of attributes unique to each process that differentiates the processes in a functional way. This provides a strong framework and context for understand the processes, and categorizing them accordingly.

The most obvious of these differentiators is iteration. The waterfall processes tend to have the least iteration, with the exception of the evolutionary processes, which have iteration in the late phases. On the other hand, spiral development is an iteration-centric

development process. Incremental development processes have an inherent level of iteration, as successive deliveries require a review of many of the design decisions. The scope of the iteration tends to be less than spiral development, but more than the waterfall and variants.

Another differentiator is the requirements process. The only process with highly flexible requirements, or for that matter, minimal initial requirements, is the spiral process. Incremental processes delay some requirements for future increments, and waterfall processes typically have the most up-front requirements, and typically seek to minimize requirements changes. At the same time, the level of feedback from prior increments or deliveries into new requirements differentiates these processes.

In addition, the level of cost and schedule's affects on the performance requirements differentiates programs into the level of design to schedule and budget use. Among waterfall variants, there are a number of differentiators as well. One of these is the rigidity of the stage-gates or milestones. The number of activities in parallel also indicates the direction of the process.

Figure 16 notionally indicates some of these differences and shows where each of the aforementioned processes stand on some of these attributes. Specifically Figure 16 identifies the processes and the attributes that significantly define them.

	Waterfall	Parallel Waterfall	Overlapping Waterfall	Design to Sched./ Cost	Spiral	Evol. Delivery/ Prototyping	Incremental Delivery
Useful Interim Capabilities	No	No	No	No	Yes	No	Yes
User Feedback Significantly Impacts Final Delivery	No	No	No	No	Yes	Yes	Yes
Planned Iteration Between Phases	No	No	No	No	Yes	No	Yes
Multiple Parallel Designs	No	Yes	No	No	No	No	No
Flexibility of Requirements	No	No	No	No	Yes	No	Some

Figure 16: Process comparison

These attributes were addressed throughout the chapter, but the figure provides a valuable reference to quickly identify the basic elements of processes. This proves to be very valuable in categorizing real-world processes, which often use variants of these processes or are cobbled together out of multiple aspects of each of these strategies. Furthermore, the attributes provide a functional definition of the processes that is more accessible to those unfamiliar with the specific theoretical processes.

Though these processes address a number of risks and uncertainties, it is important to understand that in practice there are a number of interpretations of these, and that many actual processes are comprised of different aspects of each of these processes. Accordingly, understanding these processes, but more importantly their attributes, provides a better understanding of the product development strategies available to managers today.

3.9 Product development summary and key variables

The literature review helps to expand the questions established in the introduction. As a reminder, these are:

1. What knowledge is necessary to adequately select a product development strategy?
2. When does Evolutionary Acquisition make sense? When is the spiral process applicable? When do other strategies have greater value?
3. How can the acquisition community best implement these various processes, particularly Evolutionary Acquisition?

In order to select a product development strategy that best suits the program, the process must address the various aspects of the development. Specifically, these involve aspects of the product, development environment, and stakeholders. Accordingly, the hypothesis for this research is that the variations in these aspects will match best with different development strategies. In addition to this, process analysis must be performed.

The most crucial part of this analysis is an understanding of the product development strategy. This is fundamental to how the product is developed and determines the course the program takes. Processes vary in a number of ways, including iteration, flexibility, deliveries, and process drivers. Key questions include:

- What is the level of iteration in the program?
- Are multiple incremental capabilities provided?
- How do requirements evolve in the program?
- What was the influence of cost and schedule on the program?

The product itself is a key part of how the development process is selected, as well as executed. Though products are numerous, it is possible to categorize products as well as identify them by key attributes such as architecture. Furthermore, the product intent and strategy are crucial to the development lifecycle. The underlying technologies of the product are factors in the ease of development as well.

- What type of aerospace product is being developed?
- Is the product a new development or a derivative of existing products?
- Does the product use a modular or open architecture?
- What portion of the product's functionality is implemented through software?
- What is the extent of new development work required?
- How does the rate of change of the underlying technologies compare to the product?

The program practices were also identified as crucial to product development. These include the use of system representations, stakeholder identification and involvement, and testing practices. Key questions include:

- What is the level of involvement of key stakeholders in developing the acquisition strategy and system requirements?
- To what extent were system representations used, and how did their use influence the development process?
- What was the level of combined developmental and operational testing?

Another important aspect of the hypothesis is the way in which uncertainties affect the outcome of the product development process. In the case of the Air Force these include:

- What has been and is expected to be the uncertainty in funding?

- What is the certainty in operational requirements?
- How likely are goals for performance-critical technical advances to be achieved?
- What is the likelihood of technology obsolescence?
- What is the likelihood of changes in the operational environment?
- How easy is the system to adapt to changes?

Specifically in addressing issues around spiral development there are a number of issues brought up in the literature review that are crucial to implementation. These include resource allocation and user funding commitment. In particular the following questions were discussed:

- In what areas did the program require additional resources? In what areas did it require less?
- Was the user committed to funding a spiral development process?

These questions provided the basis for the research strategy of survey and cast studies discussed in further detail in the next chapter.

4 Data gathering and research methodology

The previous sections have addressed some of the issues relevant to product development and acquisition strategies, but in order to assess the current state of these processes, and to highlight the context of the strategies, it is imperative to gather the necessary data in a satisfactory manner. This includes the approach of the research, what methods were used to gather data, what was done to ensure the validity of data, and finally how the data were analyzed.

4.1 Approach

Chapter 3 concluded with a list of questions that have not been satisfactorily answered in research to-date. To answer these questions, and make a substantiated recommendation for implementing Evolutionary Acquisition, data was to be gathered in two strategic ways. The first of these was a two-part survey, intended for system program directors (i.e. those who managed a portfolio of programs), and program managers (those managing a single program). The second method was a series of case studies directly examining those programs with Evolutionary Acquisition and/or spiral development strategies.

The categories looked at were established in the literature review in Chapter 2 and 3, along with feedback from program managers with experience in Evolutionary Acquisition. These include:

- Product architecture
- Product strategy
- Technology
- Product development process
- Uncertainty
- Stakeholder involvement

4.2 Data Gathering

The nature of the key questions raised above lends itself particularly well to a survey approach. Specifically, the research asks what and where questions, which are commonly associated with surveys. This confirms much of the precedence established by prior research in product development. In the attempt to evaluate where military adoption and practice of Evolutionary Acquisition stands, it is important to 'cast a wide net', so to speak, and take a snapshot of the system today. Accordingly, a wide survey of practice would both help the acquisition community by providing them with helpful insight into what they are doing. Secondly, the surveys provide the raw data needed to correlate practice with product, risk, user, and customer. However, in terms of a developing a better understanding of Evolutionary Acquisition and spiral development, it is valuable to capture the experiences of program managers who have implemented these processes.

Surveys

In order to deliver a tool for project managers to effectively pick the 'best' product development strategy, it should be based on the knowledge they have available, as well as a means of evaluating that knowledge.

What information then is necessary? Two prime categories are the data known to the project manager in the front-end of the development, and the data that are necessary to correlate that knowledge to the processes, in addition to the program outcomes, to determine the success of the programs. When broken down further, these comprise the four subjects mentioned above—product, process, risk, and user needs. Of these, risk is perhaps the most encompassing category, as there are product-specific risks and user-specific risks in addition to the traditional

risks identified before. Within the program director's realm, it was important to address the associated management topics. Accordingly, two surveys were identified—the longer survey to look at specific programs and product acquisitions, and a short survey at the program director level to look at portfolios of acquisition programs. The surveys, which expound upon the key questions and constructs found at the end of the literature review, can be found in Appendix A.

Case studies

The questions in the case studies were culled from questions in the surveys, and were designed to be broader and require more in-depth responses than the survey questions. These questions are in Appendix B. Specifically, the goal was to better understand what some of the key enablers and challenges in implementing spiral development were. The questions were culled from the survey questions and key constructs identified through background research. They were specifically intended to address more of the practices and challenges of program managers, as well as the uncertainty in the development process.

All data was gathered through programs acknowledged by various program directors and the Acquisition Centers for Excellence, both at the Pentagon and at the bases, as programs that had experience with a spiral or evolutionary development strategy. At the same time, these programs were identified as successfully implementing Evolutionary Acquisition, and do not necessarily represent the experience of all programs implementing these strategies. Attempts were made to gather as wide a variety of programs as possible, and to keep burdens on the programs minimal. The result was an interview process involving both a program manager and chief engineer in most cases, to discuss programmatic and technical issues, respectively. Interviews averaged ninety minutes, not including follow up questions later. At the same time, the majority of

interviews were performed in person at the program offices. Participants were voluntary and were informed of that all answers were optional, and that no program names or participants would be identified. Details of this can be found in Appendix C, the consent form for studies.

4.3 Research realities

As stated above, the initial intent of the research was to survey a wide variety of aerospace development programs, each with its own development strategy—the population for study comprised all USAF acquisition programs. Though the survey was completed, due to unforeseen circumstances, it was not possible to gather the data necessary for an analysis of the state of acquisition in the Air Force. The survey process did feed directly into the case studies, as pre-testing of the survey with program managers with spiral experience helped to expand the author's understanding if real world spiral development. Furthermore, the survey also helped to clearly establish the metrics and framework for the research. The result was that the case studies, which had initially intended to be used as supplements to the survey research, became the sole source of data. This was a return to the early stages of this research project, when the author had been tasked with evaluating and understanding spiral development rather than evaluating product development strategies as a whole. This followed the goals of the Pentagon's Acquisition Center for Excellence (ACE), which was determined to better support the new Evolutionary Acquisition.

The survey results will eventually be released as a paper through the Lean Aerospace Initiative at the Massachusetts Institute of Technology.

Despite the inability to use the surveys as a direct part of this research, the development of the surveys was beneficial in itself to an understanding of product development strategies. Furthermore, a better understanding of Evolutionary Acquisition and spiral development was still possible through these studies.

5 Case studies in Evolutionary Acquisition

The case studies performed as part of this research exhibited a number of unique characteristics, but at the same time, there were a number of similarities that provide a valuable set of lessons for programs attempting Evolutionary Acquisition in the future, as well as for policy makers.

To place these various programs in context however, it is valuable to look at an example case study on a historically successful evolutionary program, the F-16. In particular, the example should provide the reader with a better understanding of development practices, timelines, and historic illustrations of evolution. There are a total of 7 case studies including the one historical case study on the F-16. This chapter provides an in-depth look at each of these studies, focusing on the program's acquisition strategy, the various uncertainties encountered or expected in product development, key enablers for program success, and major challenges encountered by the program. These last two sections provide a valuable set of lessons learned for each of the programs, while the acquisition strategy places the development process in the context of the literature review and the theoretical processes.

5.1 F-16 Case Study[1]

Program Background:

The F-16 is arguably the most successful Air Force program in the last 30 years. Over 4000 aircraft have been delivered to date, and backlog extends well into 2009. With over 9 million flight hours and a 71-to-0 air-to-air combat record, the aircraft is a performance success story as well. The story of how the F-16 was developed, and later how it has adapted, is extremely relevant to the current Air Force strategy of Evolutionary Acquisition.

[1] Case study adapted from paper by Capzzoli, P. et.al. (2002) The author of this thesis also co-authored the original case study.

To begin, the F-16 emerged from the Lightweight Fighter Program, which was a direct attempt by U.S. Forces to counter growing numbers of Soviet fighter wings in Eastern Europe. Essentially, the goal of the program was a lightweight, and therefore low cost, fighter that could be fielded *en masse* against the Cold War enemies, and would compliment the existing Air Superiority Fighter, the F-15. It was a new strategy of warfare, and the Lightweight Fighter aircraft were essentially the first of their kind, and were not intended to replace an existing system.

Following a Request For Proposal (RFP) in early 1972, contracts for prototypes were awarded to the YF-16 and YF-17 (Northrop). First flight was in 1974, and by 1975, the YF-16 had been declared the winner of the LWF program. 1976 marked the rollout of the first Full Scale Development (FSD) aircraft, followed in 1979 by the F-16's entering service. Subsequently, the aircraft has gone from the early A/B models to C/D models, and numerous Block Upgrades have been performed (the program is currently in Block 60 development, with Blocks coming in intervals of 10 after Block 30, and in intervals of 5 from Block 1 through 25).

Acquisition Strategy:

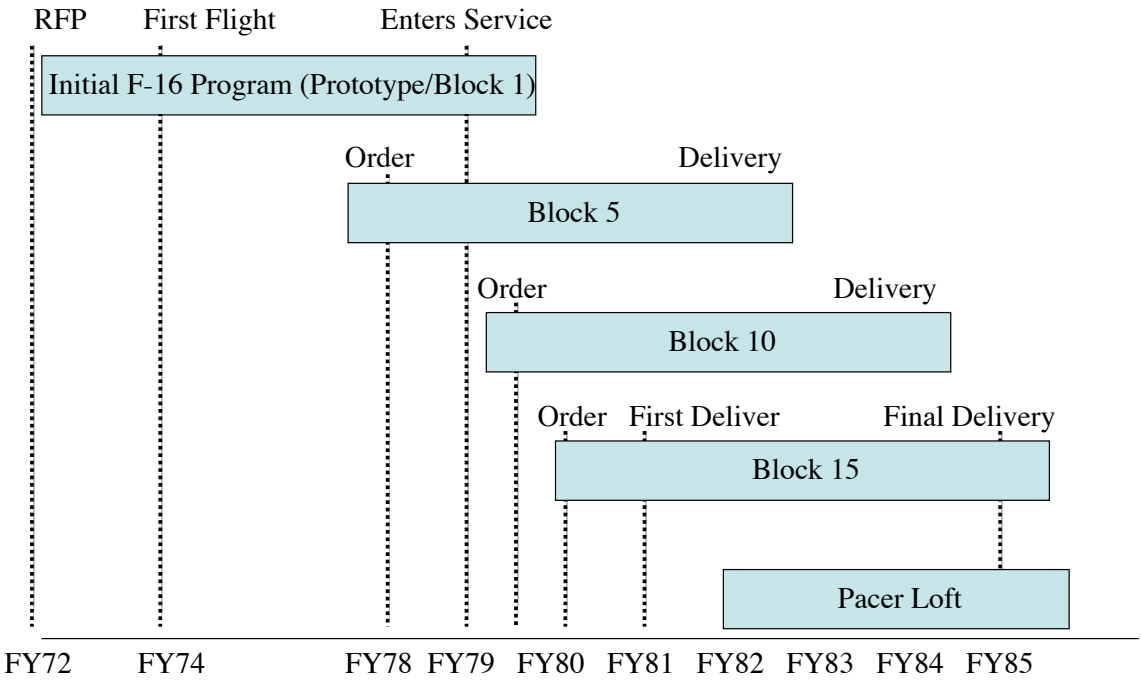


Figure 17: F-16 A/B Acquisition Strategy

The first FSD models (Block 0) had enlarged aerodynamic surfaces to cope with the additional loads expected as the plane transitioned into a multirole aircraft, as well as a operationally useful radar. Versions A and B (single and two seater models, respectively) continued with Blocks 1, 5, and 10, which were primarily improvements in reliability. Block 15, the most numerous production block, strengthened the structure, once again increased aerodynamic surface, and added capability for new munitions, essentially increasing the multirole capability of the aircraft. Note the timeline of the strategies shown in Figure 17. From block inception to completion typically took on the order of 5 years, and new blocks were ordered well before previous blocks were complete. The Pacer Loft program brought the earlier Blocks up to Block 15 standards. Block 25 represented the first of the version C/D aircraft, with significant avionics and cockpit upgrades, including a vastly improved radar system, AMRAAM capability, cockpit multifunction displays, and it was the first F-16 series to feature night and precision ground attack capability. Blocks 30/32 (same basic aircraft with a choice of Pratt & Whitney or General

Electric engines) had a larger inlet for the new GE engines, and further munitions capability. Block 40/42 was a improvement intended for night/precision and all-weather attack missions. Once again the structures were upgraded, and capability to use Low-Altitude Navigation and Targeting Infrared for Night (LANTIRN) systems was added. A newer radar and heads-up display were added along with a terrain following mode and capabilities for additional bombs. Blocks 50/52 took over the Suppression of Enemy Air Defense (SEAD) role and added full autonomous anti-radar capability through the use of external pods. Furthermore, the system incorporated a new modular mission computer. The most recent Block 60 aircraft still in development are intended for the United Arab Emirates only, and will feature new engines, new radar, and will internalize many of the features the F-16 was previously only capable of with external pods.

Most interesting is the fact that the initial program took the bulk of the work, while upgrade programs typically took only a couple years of development (and additional time for delivery). Nevertheless, looking at Figure 17, it is evident that development on new blocks was not only going on before previous blocks had been completed, but also even before previous development had completed. In this way, however, the program was able to continuously improve the product's performance, made easier by the fact that the platform already existed. Essentially, there were numerous F-16 upgrade programs running at the same time.

Key Enablers:

To implement the low cost and high performance goals of the competition, General Dynamics (now Lockheed Martin) determined that advanced technologies would only be used where they were warranted—i.e. where there was a high expected return. At the same time, throughout the

competition, the developers were aware that markets in addition to the Air Force, such as NATO allies and others, would be keenly interested in acquiring a low-cost fighter with varying capabilities. The result was a key decision to use a modular architecture from all aspects—airframe, avionics, engines, and so forth—so that the aircraft could be upgraded in the future. The program was given no real requirements, with the exception of two goals—low cost and high agility. Specifically, there were several key maneuverability targets, but beyond this, programs were left to their own.

During the development, the program used several practices that are still excellent by today's standard. In response to the proposal, all programs were to bring aerodynamic models to the initial competition for prototype contracts. Following selection, the YF-16 program itself did a number of things that are still recognized as good practices. On their development teams, they included Air Force members—two pilots, and several people from maintenance, reliability, and manufacturing—but as a direct result of the program manager, the program office was limited to 10 members, tasked with understanding the program, but not interfering. The YF-16 was one of the first programs to use a cross-functional or Integrated Product Team (IPT). In addition to this the organization was arranged to be a matrix organization, emphasizing both program and functional management.

In retrospect, the initial program was heavily constrained by cost and time, and it is clear that the program had to prioritize the design goals. This led to the philosophy of including only those technologies that had the greatest payoff, while leaving room for later improvements. Furthermore, despite using weight as the metric for cost estimation, the program made the cost

information available to all engineers, and trade-offs were based on this. In addition to this, the Chief Engineer was adamant about quickly and thoroughly completing the preliminary design to minimize the risks of major changes down the road. At the same time, the program used prototypes, particularly in the human-machine interface area (the cockpit), to evolve this interface for development. When the first prototypes were complete, the program used these to evaluate the Air Force's needs, and a number of changes were made from the prototype aircraft to the production models. The testing itself, due to the lack of requirements, was effectively a test to determine the operational capabilities of the aircraft itself.

The F-16 has gone through a number of performance upgrades over the course of its lifetime, and will continue to evolve until it is replaced by the F-35. Though the initial program succeeded as a result of its good program management as well as technical capability, subsequently the program has succeeded because of its ability to adapt to new roles and capabilities with relative ease, while keeping costs down. This is primarily because of the decision to have a very modular product, from the airframe manufacturing design, that allows for changes to individual sections, to the use of modular avionics, both through the ability to upgrade internal systems as well as add functionality through the use of external pods. This allowed for ease of changes to the system, which in turn allowed the system to keep up with changing times.

Summary:

In essence, this program exemplifies what the Air Force wants from its programs today. Like the situation during the LWF program, the Air Force is faced with ever increasing costs of procuring new technology as well as product development cycle times that have expanded to over 10 years

between concept and delivery. With Evolutionary Acquisition, and spiral development, they are trying to do what the F-16 did—quickly deliver what they can, and evolve the capability over time. This follows the military variation of incremental development, incorporating some flexibility in blocks, but also establishing the platform for future upgrades early on. At the same time, because of the number of simultaneous F-16 programs, the warfighter was able to quickly get new capabilities.

5.2 Research case studies

The following case studies of current systems looked at four essential areas: technical aspects of the product, program resources, the perceived variation of the development environment, and the acquisition strategy. While each program differed in all of these areas, there were many shared experiences across programs and a number of unique but valuable practices in each case.

The cases have been disguised to protect the programs and people involved, but the essential information required to better understand Evolutionary Acquisition is still there.

5.3 Evolutionary Case Study 1 – Program A

Program Background:

Program A started in May of 2002, with the development of the first increment of useful capability, followed by a second increment begun in March 2003. There are a total of eight increments planned, and the program is expected to continue past 2010. The program primarily focuses on database reengineering, and is software dominated, with hardware serving a supporting role. In addition, the program is replacing a legacy system incrementally, while improving on the system performance. At the time of this writing, the first increment had been delivered to the user, while the second increment was in development.

Acquisition Strategy:

The program follows a strategy of incremental deliveries comprised of smaller spirals, shown in Figure 18.

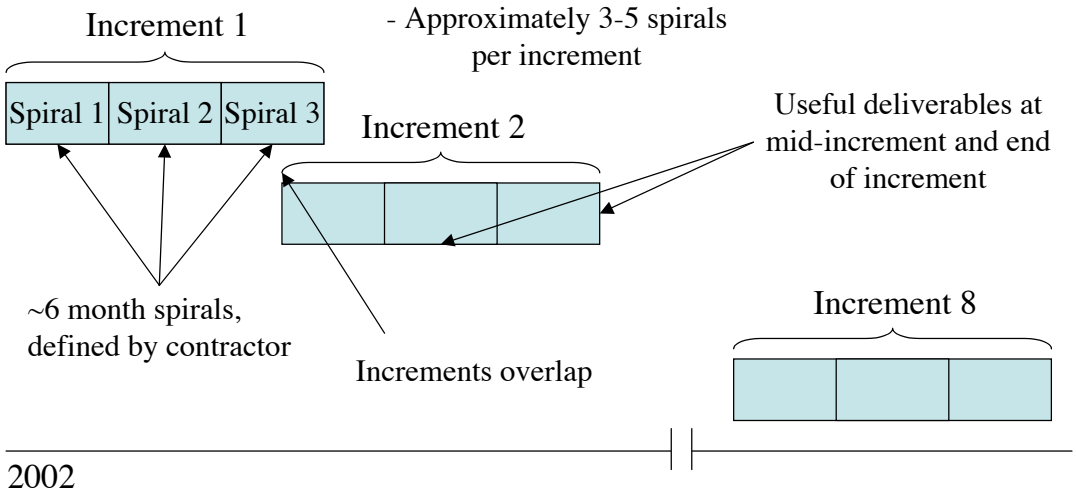


Figure 18: Program A acquisition strategy

Increments occur on a twelve to eighteen month schedule, and spirals grow the capability until the increment threshold is met. Spirals are less than six months long, and a typical increment consists of three to five spirals. Increments also overlap, and are defined by the Technical Requirements Document (TRD). Each increment typically yields two deliverables, one midway through the increment and one at the end. Spirals are not defined by the program office, but rather by the contractor, and typically consist of demonstrators for evaluation by the program, to be used for evaluation.

Programmatic Uncertainties:

The program did not encounter any significant uncertainties so to speak, though it did have trouble in the first increment (to be discussed in more detail), but these were not particularly related to uncertainty. This program prioritized the key design tasks and requirements for each increment, so that there was little difficulty in achieving the established operational requirements. Changes to these requirements were made through a Configuration Control Board, which reviewed both the pertinence of the changes as well as the involvedness of changes, to determine whether requirements changes were necessary in that particular increment, with the potential to move changes to future increments. As a result of the two-fold requirements process, the program experienced little unplanned operational requirements changes, and expected the same in the future. The program did not encounter any significant technical advances through the first increment, and due to planning for technology refreshes in later increments, was not concerned about technology obsolescence. In particular, the product consists of servers and software, where many standards exist, and allow for flexibility well into the future. Budgeting, which is traditionally a significant concern for program managers, did not appear to be a significant challenge in this program. Specifically, the program manager cited the

requirements prioritization as the key to this. Not only was the prioritization used to understand what was possible given funding in a specific increment, but also provided flexibility down the road if and when budget challenges were encountered.

Key Enablers:

Program A had a number of enablers that allowed the program to proceed with a smooth spiral development. While some of these were planned in the program, several of the enablers emerged naturally.

First and foremost, the program's users had experience with spiral development on prior programs. This undoubtedly led to a much better understanding on the user's end of what was realistic in the increment goals and what was expected on the part of the user. In terms of the testing, the program was able to use the actual facilities that would be using the product, which eliminated the need for external test facilities and associated work. In addition to the fact that the program was a primarily software solution where demonstrators are somewhat easier to develop, the hardware used, such as servers, were relatively mature. Furthermore, the system is organized around several distinct tasks, including data gathering, access, analysis, and prediction, such that there were few interdependent tasks. The chief engineer of the project noted that the functions had very clean and clear interfaces. This facilitates the use of a modular architecture, and accordingly, simplifies changes to the system. From a programmatic standpoint, the program manager noted that cost estimation in this program was much easier as the planners were able to use historical data to estimate the costs.

While the program had some innate enablers, several key decisions on the part of the program office and key stakeholders were made that further allowed for a successful development project. The first of these decisions was on the part of the key stakeholders, primarily the user, to be heavily involved in the requirements development process. In part due to this, and also as a result of prior experience with spiral development, the user and developer could agree on realistic increment goals. Furthermore, capabilities and requirements were "racked and stacked," allowing for a clearer understanding of what requirements were going into individual increments. From a programmatic standpoint, specifications are flexible as well, and can be changed even after a review. At the same time, the program has been flexible in incorporating program and product lessons learned from the use and evaluation of increment 1 into increments 2 and 3, though more so into the latter. In the product decisions, the program elected to use an estimated fifty percent non-developmental (primarily Commercial-off-the-shelf) hardware, though more could have been used were it not for the requirements on legacy systems. By doing this, the program has ensured that the system evolves at a similar rate to available technologies.

Program Challenges:

Though Program A has been a successful program, "a model of spiral development" according to one ex-program manager, they did encounter a few challenges, some of which are generally applicable to spiral development programs in the Air Force, and some of which are more unique to the program itself.

The largest problem the program encountered was in the first increment. The issue appears to have been with the developer's inability to deliver the desired capability. Specifically, the program manager cited several key issues, including failure to quickly deliver capability and

technical and managerial challenges within the developer. In addition to this, the program felt that the pace and requirements for the first increment exceeded the contractor's capability. As a result, the program fired the contractor. Subsequently, the program office re-scoped the first increment and hired a new contractor, moving those tasks that were eliminated from the first increment to the second. In addition to these challenges, the manager noted that there is a significant burden placed on the program management in both working on the current increment as well as having to plan for the next increment. This is in part due to the fact that increments overlap each other. Finally, the chief engineer noted that from a design perspective, there were some challenges with the use of a legacy system, which limited the number of design options. Essentially, the program was driven to a smaller set of options due to some interface requirements with somewhat obsolete technology.

Summary:

Program A exhibits several characteristics that make it a valuable example of spiral development. The first of these is the spiral within incremental delivery. In it, each increment is delivered with feedback occurring within the increment though the use of demonstrators. In addition, the program was able to take advantage of its natural division of functions to use a modular design. Program A also had good buy-in from its users due to the fact that they had prior experience with spiral development and also were able to prioritize design tasks successfully. This allowed for very reasonable goals within the increments, and a clear understanding of what the outcome of each spiral and increment would be. At the same time, the program encountered some difficulty as the first contractor was unable to deliver at the expected pace. Within the program, there are challenges with management resources, having to both manage the current increment and plan for future increments.

Program A's acquisition strategy was a hybrid of several of the theoretical processes. First of all, it used a combination of incremental and spiral delivery. Though increments were loosely defined, the program grew capability in these increments through spirals. Furthermore, increments were not completely defined up front, but were open to reasonable requirements changes. This program looks very much like a block upgrade strategy where blocks are spirally developed in themselves.

5.4 Evolutionary Case Study 2 – Program B

Program Background:

Program B began in October 1998, and fielded its first of five increments in March 2002. It is a software program, essentially providing a dramatic improvement and new capabilities over the existing system, and is built from scratch, using about 50 percent commercially available software. The software was initially intended for use by a small group of users, but this number swelled to over a thousand during the initial development phase. A complex system with over one million lines of code, the software uses databases and provides significant planning capabilities. It is run on an existing secure network, and is integrated with existing software systems in use by the Air Force.

Acquisition Strategy:

This program uses two-fold release process of major increments delivered every couple years, with smaller spirals, or emerging releases fielded on the order of every six months. Figure 19 shows the process graphically. The program currently calls for increments to be delivered again

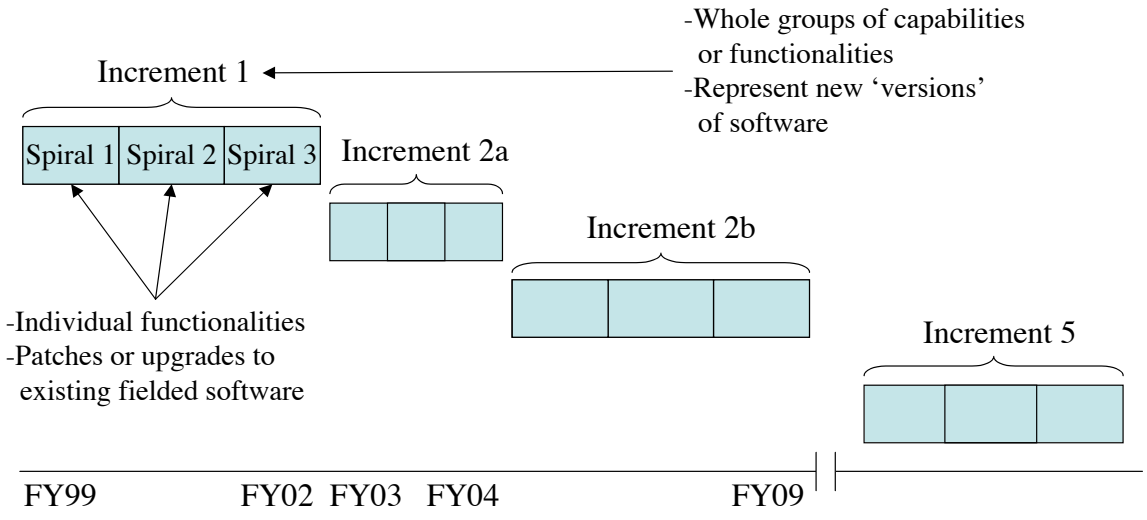


Figure 19: Program B acquisition strategy

in FY04 and FY08. Essentially, increments contain whole groups of capabilities or functionality, while spirals may contain one or more of the individual capabilities. From a traditional software standpoint, the increments are new releases of the software, similar to a new version, while the spirals are patches or upgrades to the fielded software. At the same time the spiral patches are fielded. Increment 2 and 3 are underway, while only minimal scopes for Increments 4 and 5 are known.

Programmatic Uncertainties:

Program B established several guidelines to reduce its risk. The first of these was a clear understanding that requirements changes within an increment would not be accepted without the additional funding to make it feasible. The program has had good success with this policy on the first increment, but some difficulty on successive increments, as the functionalities provided in the second increment are more integral to the system capability, and these requirements cannot be sacrificed as easily. While it has reduced the creep in operational requirements, it has not reduced budget cuts and associated capability cuts. From a technological perspective, obsolescence is expected to be a problem by the end of the program in large part because of self-inflicted choices, particularly the programming language. In addition to this, new advances in technology pushed the program to adopt web enablement and portals as well as new user tools. One of the bigger challenges in this program was the change in operational environment. In the time between program initiation and delivery of the first spiral, a new user group was established that had not been anticipated, and loads on the software were significantly higher, requiring some changes and support by the program.

Key Enablers:

Program B has had a great deal of success in involving the user and other key stakeholders throughout the program. As in many programs, each spiral or increment must receive user approval through user representatives on two-tier IPTs, called user advisory groups. These groups submit recommendations to the contractor on a daily basis, with the stipulation that these cannot be baseline changes to the program. In addition to this, the results from the first increment fed directly into the following increment. In fact, Program B is the only program in these studies to not immediately overlap increments. In particular, the user realized a greater need for the program's services, leading to some new capability requirements in increment 2b. In addition to this, the program has had good involvement with testers, and has procured dedicated test suites, reducing some of the challenges typically associated with the multiple tests required in spiral development.

From a programmatic perspective, Program B has elected to use Cost-As-an-Independent-Variable (CAIV) as a means of determining what capabilities and requirements will be in future increments. What's more, the program has been successfully able to move non-mission critical issues into successive spirals and increments.

Program Challenges:

Despite heavy involvement from stakeholders, Program B has had a number of challenges with managing expectations, primarily in terms of funding. Primarily, the funding line does not support spiral development. This manifests itself in several ways. First of all, the funding fails to address the fact that the deliverables likely will have some problems, so typically money is routed from future increments to repair and support the latest increment. In addition to this, though the program has set aside time between the spirals, for generating experience and

providing support, the funding does not recognize that this period has associated costs. Furthermore, the budget did not support performance testing. Additionally, the program's cost estimate was likely to increase as increments were defined in more detail, such as the case of increment 2, which will cost about three times more than was initially authorized by the budget authority. In general, the program manager noted that the issue was primarily with moving increment plans from the macro to the micro level. The manager also noted that the user did not clearly understand that changes were costly, and that baseline requirement changes in future increments could not be performed without funding. Beyond budgeting issues, resources were not available to work on future increments because current challenges took priority. Moreover, there were a number of other stakeholder issues. Though the Air Force gave tutorials to both the developer and the users, initially the user did not understand the difference between spiral and traditional programs, with regard to the differences in deliverables. This also created a challenge in coming to a common definition of what was to be in each increment and spiral. Accordingly, neither the contractor nor the user understood the complexity of implementing the solution—not only replacing the existing system, but providing significant new capabilities, and hence a significant change in the business rules. In fact, the program's Concept of Operations (ConOps) is out of date, and the program is applying policy not yet approved by regulations (though these are in the midst of changing).

The program also had many design constraints. In addition to the nearly two thousand-fold increase in usage over the program's initial anticipations, the requirement on interoperability and use of key technologies made this program a key candidate for spiral development, but also created a number of challenges. Though new technology requirements such as the use of web-

enablement and portals was significant, the larger challenge has been in coexisting with military-specific systems, many of which are going through similar upgrades and version cycles. This places the additional burden of maintaining interoperability on top of delivering new capability. This is magnified by the fact that the program software is highly coupled with these external systems. In fact, all of increment 2a is essentially devoted to interoperability upgrades. Accordingly, the program is very sensitive to external changes, and internal changes can adversely affect outside programs. Nevertheless, the program is helped by the fact that it is modular within itself.

Despite the dedicated test facility, there were a number of testing issues that the program manager brought up, specifically dealing with spiral development. The most obvious of these is the requirement on testing for each spiral, resulting in the hiring of two additional staff members for testing. While many of the tests are similar, being the same type of test and having similar duration and depth, due to the coupled nature of the external systems, integrated testing is often a scheduling challenge. Another issue with the frequency of releases is that performance tended to be a stronger driver than schedule, which conflicted with the desire to field new capabilities quickly. This was further hindered by regulations that created a significant barrier to releasing new versions, so the program was often forced to use patches, creating a number of maintenance and logistics issues.

Summary:

Perhaps as a result of the challenges in implementing spiral development, Program B serves as a very valuable guide for acquisition. From a testing perspective, though the use of a dedicated facility was helpful, it was clear that spiral development generated a number of other issues in

testing, particularly due to the frequency of releases. In addition, the interoperability with external interfaces proved to be a challenge to the program, requiring significant resources. One of the unique points of Program B is that it intentionally set aside a period between increments, though this was not supported financially. In fact, a great many of the challenges encountered by this program had to do with managing expectations and coming to a clear understanding of what was required of each stakeholder. From the program's perspective this had primarily to do with requirements versus funding, but this in itself was due to the inability of the stakeholders to understand the impact of funding cuts on system performance, in addition to the program's inability to properly estimate costs up front. Perhaps the most useful lesson from this program is to clearly establish the expectations from all stakeholders and as the program manager, to manage these expectations.

Program B also was the program that seemed closest to a spiral program, as there was a period between increments for learning. At the same time, it used a mixed spirals and increments approach like Program A, indicating a block upgrade strategy with miniature spirals. Program B seems to have less of the design-to-schedule or budget strategy of Program A, particularly as the program manager stated that the performance goals for the spirals and increments was more important than the schedule.

5.5 Evolutionary Case Study 3 – Program C

Program Background:

Program C is a two-part program involving what are essentially two systems, designated as System C-1 and C-2 here. The overall program involves a mix of hardware and software, but hardware is primarily off the shelf, standard equipment, while the software is the main focus of the program. System C-1 delivers a ground-based platform for improved surveillance, and uses existing infrastructure, while C-2 will be a portable version of this. While there are two systems, C-2 is an offshoot of C-1 and will use much of the software developed for C-1. C-1 began in April 2003, and C-2 will begin in early 2004. The program is expected to continue through 2009. The primary focus of this study will be on System C-1, as the program has reached the development phase.

Acquisition Strategy:

There is an individual acquisition strategy for each system, though they are not unique, and there is a significant relationship between the two strategies. Both systems use spirals to deliver capability, as shown in Figure 20. In the case of C-2, the spirals are called blocks to prevent confusion between the two systems. The key here is the common software that will come out of C-1's spiral 2 and feed directly into the first 'block' of C-2. In addition to this, spiral 1 will deliver two development systems to sites in June 2004, and three production systems in late 2004. In the case of system C-2, the first block will focus on developing a hub for the subsequent block. As shown in Figure 20, system C-2 will have multiple deliveries throughout the second 'block'.

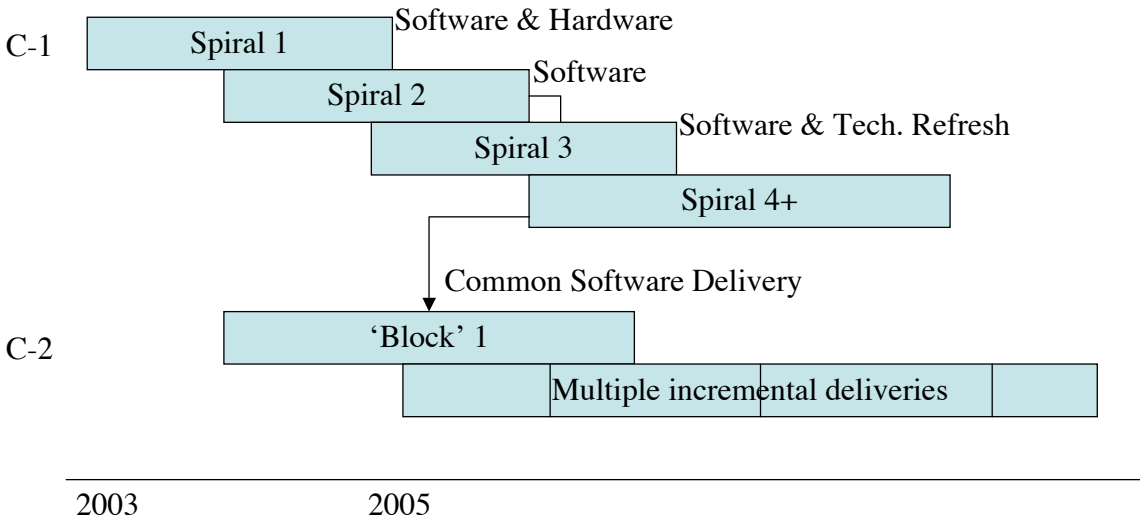


Figure 20: Program C acquisition strategy

Programmatic Uncertainties:

While many of the programs examined in this research indicated that budget cuts were always an issue, Program C was very confident in it's funding. In response to concerns about operational requirements, the program responded in much the same way that other programs have, by prioritizing requirements, and executing those requirements that were deemed most necessary first. Due to the pre-existing infrastructure of C-1, there was little worry that there would be operational environment changes, but for C-2, this was not the case. Particularly, there have been a number of discussions on the use of system C-2 in places it was not originally intended for, which would require further development. The program manager for both systems realized that technological obsolescence would be an issue, and planned for technology refreshes every three years. The issue here, he noted, was that this created not so much of a development problem as a logistics problem, as typically spares are kept on hand for longer than this period, and there is little need for the number of spares to currently be acquired if the technology is being refreshed.

Key Enablers:

Due to the complex nature of Program C, there are a number of very interesting findings that provide insight into the Evolutionary Acquisition strategies employed in the Air Force.

Program C had initially been a traditional program that was converted to a spiral development. In particular, the events of September 11, 2001 provided a mandate to the program, along with thirty million dollars and the order to "get what you can." Their acquisition strategy now specifically calls for evolving capability. From a program perspective, as a result of the instructions given to them, they were a design-to-cost program. To begin the spiral development, the program worked with the users to determine 464 requirements and prioritize these into top priority and lower priority tasks, with the understanding that the first spiral would be the most important. Specifically, the program managers noted that the first spiral must provide adequate capability, sustain legacy and new equipment, and must satisfy the user so that they would both accept the product and continue with the spiral process. In addition to this, the program had a strict policy of not adding requirements to the spiral. The result was of the 464 requirements, 404 were delivered in the first spiral, of which 288 were top priority. The managers noted that executing to schedule and performance was a less of a challenge than executing to cost. Specifically, they noted that while the traditional system of acquisition was dominated by addressing cost and performance, the focus has shifted towards schedule.

From a technical perspective, there was little development on the part of the program. Like several of the other spiral programs, the key issues seem to be mostly in integration. This is a result of an estimated eighty to ninety percent use of Commercial-Off-the-Shelf or non-developmental technologies. At the subsystem level, no single subsystem was more than forty

percent new development. The program manager also noted that there were not a lot of interoperability and standards issues beyond basic issues, likely due to the fact that system C-1 has a very set infrastructure. The product is relatively modular, but certainly not 'plug-and-play'. This could change however, as there are several Air Force initiatives underway that would change the operational environment and require the use of specific technologies.

Program C also has significant use of testing, and a very close relationship to testing. A notable achievement was the reduction of interoperability testing from one year to six months. They have had more frequent interaction for testing and coordination with their own test squad, and testing is always running, in their own test shop. They also buy their own test assets and use them full time. Specifically the program established an experimental center where they can perform new Concepts of Operations (CONOPS). Additionally, the program has the infrastructure to test and experiment with new technology. This was largely a part of the prototyping process, using a virtual reality simulator and developing a prototype that was an approximately eighty percent solution. The key, as one manager pointed out, was focusing on the prototypes as capabilities and not solutions. Otherwise, the program runs the risk that the user will ask for the specific system in the prototype rather than a system that provides that level of capability. This was a very valuable process in incorporating user experience, as well as getting political buy-in. Furthermore, the program was able to better understand many of the human-machine interfaces through this process, feeding directly into the development process.

One of the most important things the program has done is to manage expectations for the various stakeholders. Specifically, the program manager noted, the expectations were very different for

traditional stakeholders compared to political stakeholders, i.e. the real warfighter versus the Pentagon. Part of this was solved up-front, by involving the various communities and especially the user community in the requirements process, but another aspect of this was the use of 'agents' so to speak. Essentially, the program sought out operations people who supported spiral and embraced them. These people would then take the message to the rest of the community, effectively being 'agents' of spiral. The inability to manage these expectations would result in lost funding and resources, warned one manager.

Program Challenges:

If Program C's use of good practices was a result of necessity in such a complex program, then many of the challenges they had were for the same reason. Despite many efforts to manage the challenges in product development, the program managers indicated several areas that required further work.

The majority of the challenges stemmed directly from the concurrency of the program. People were often being shared between spirals, and the team responsible for one spiral was pulled to plan the next spiral as well. The difficulty here was in keeping up the level of continuity; specifically, individual spirals would lose expertise as people moved to subsequent spirals. This would show up in the test phase, where it becomes difficult to make some of the changes deemed necessary in testing. Furthermore, the program spent a lot of time working on contracts, and as one manager stated, they need a contracts officer who is more flexible. The program used a firm fixed price contract on items that it knew were necessary, and time and materials money on unknowns. One notable drawback is that firm fixed price contracts are less flexible than cost plus contracts. An additional burden on the program was the lack of a prime contractor. This

meant that the program office had to serve as the integrator for the system, causing additional strains on the office.

Another issue with the concurrency is the failure to immediately incorporate lessons learned from one spiral into the next. The program is always off by one spiral, so learning goes from the first spiral into the third, except in the case of defects or pieces that are easily changed. This again was an issue in testing, where by the time one spiral would test, the requirements for the subsequent spiral were already set. Specifically, testing seems to always be an issue of resource. In Program C, this had to do with a couple of things. First of all, despite the test resources available, the size of the testing staff appeared to be inadequate. Secondly, the amount of retesting is an issue. This is a twofold problem, brought about by the fact that spirals must be tested again, but also because without a clear continuity of testing staff, new testers must be brought up to speed on the testing program and experience so far. A program manager suggested performing regression tests on the changes made in the future.

From a technical perspective, the program is dealing with significant quantities of hardware and software, creating a logistics problem, particularly with retrofitting systems as new spirals are completed. One of the important points that one staff member noted was that the use of COTS and open architecture systems was more difficult than anticipated by the Air Force. The use of COTS, he said, would save money upfront, but cost more in maintenance and sustainment. This is due to the rapid technology cycle times and corresponding obsolescence of older systems, especially in operating systems, where the software is not always backward compatible. In this system, the logistics trail is more difficult as the use of COTS means that the program must have

multiple baselines. As a result of the short lifespan of COTS in this system, it is essentially a new program every time the COTS products are changed. Furthermore, while the Air Force pushes for open architectures, the challenge becomes data transfer, and procuring data rights to send information between software programs. Accordingly, it is often easier to stick with one contractor rather than procure multiple components from various vendors.

Summary:

Program C is faced with a challenge due to its complexity in developing two systems but provides a number of valuable lessons for less complicated programs as well. Certainly this study has brought up the fact that COTS should be heavily considered before implementation. Also, one of the most valuable lessons from this was the ability to manage stakeholder expectations through the use of change agents. Furthermore, the program noted some resource strains as it was forced to deal with system integration as well as testing challenges.

A look at the acquisition strategy of Program C, and specifically C-1 indicates a strong incremental process, particularly since the goals for the future increments are somewhat well defined. At the same time, there is still some flexibility in these increments. The initial spiral was essentially a design to budget strategy with the mandate to accomplish what was possible given the funding.

5.6 Evolutionary Case Study 4 – Program D

Program Background:

Program D began in October 2002, and is intended to last eight years. At this point, the program has used approximately five percent of its billion dollar budget. It is a time-critical upgrade program necessitated by a change in the operational environment. At the same time, the program manager stated that the development aspect of the program was less significant than the manufacturing throughput challenges the program foresees. It is an aircraft subsystem, and is accordingly primarily hardware with a minimum of software support.

Acquisition Strategy:

The acquisition strategy employed by the program is essentially a mix of concurrent and incremental work. Unlike many of the other programs, there are no spirals within the increments. Figure 21 graphically shows this strategy. Note that the increments are overlapping

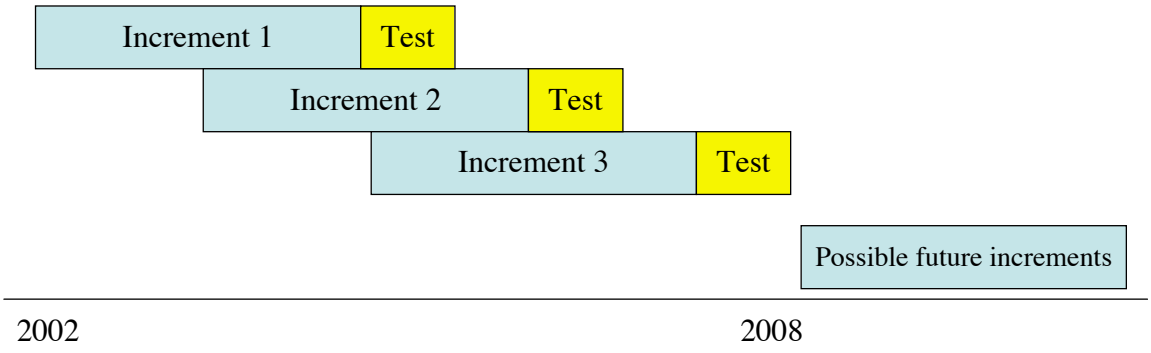


Figure 21: Program D acquisition strategy

and that test occurs at each increment's end. Each increment is fielded immediately after test. In addition there is the possibility for additional increments or spirals in the future. The strategy is essentially to go into Low Rate Initial Production (LRIP) for the first increment and prove the technology. Due to the immediacy of the program and the operational need for the system, the goal of the system is not necessarily to provide improved capability over the current system, but

rather to replace the current system with a usable one in a specified period of time. The second increment will deliver nearly 90 percent of current capability, while the final increment will bring the new system up to the performance standards of the current system.

The increments are very well defined and serve primarily to deliver technologies as they become ready. There is some feedback from flight-testing into the subsequent increments (i.e. testing of increment 1 will feed into increment 2), but primarily to address performance issues and any problems. In addition to the hardware deliveries, there are three software deliveries. At the end of the three increments planned, there is potential to enter into further development and capability upgrades.

Programmatic Uncertainties:

As Program D is specifically a program to address a change in the operational environment of the system being replace, it is somewhat ironic that the program does not expect any real changes in its operational environment. At the same time, it has several other uncertainties. From the perspective of technological obsolescence, there are a couple of concerns. First of all, the program delivery volume is very small, and the program is not procuring quantities on a level that would justify the manufacturer's continued production of the technology were it to be replaced. Secondly, the program manager noted that the number of Military Standard (MILSTD) parts have been reduced, creating a greater dependence on Commercial-Off-The-Shelf (COTS) parts and further intensifying concerns on obsolescence. In addition to this, the system must be nuclear certified, meaning it can operate in an environment with nuclear weapons, a process typically eased by the use of MILSTD parts. More uniquely, the program did not have other concerns, such as requirements changes, as the system of which the program is a part is not

changing, and the program goal is set clearly to replace the existing subsystem within a known timeframe. Furthermore, because of the immediacy of the requirements, it is unlikely that the program will face any budget issues.

Key Enablers:

The program's mandate to quickly develop and field the required technology helped immensely to determine and execute the development strategy. To begin with, the user was involved at a very early stage, and chose from the options in the trade study, essentially between modifying the existing system and developing a new system. The decision to develop a new system was based in part because of an impending Mid-Life Upgrade (MLU) scheduled for the old system, which would have been in addition to the cost of modifying the system. Logistics and testing were also brought in early as the program realized the challenges implied in this task, and in particular in the rapid and frequent testing.

From a resource perspective, the program manager felt that overall resource needs were fewer because of the concurrency of the program. In particular the short program time meant that contractors were not needed for as long, and that there would be no need for a standing army. Testing resources were significant however because of the number of test periods, although results from the prior tests could be used towards the final tests. Another key in the program was to hold the engineers to the increment goals and not exceed them, though possible. This is again attributable to the schedule pressure.

The program did not have significant budgeting issues as initial funding was inadequate. Money eventually came from improving efficiency elsewhere in the system and overall costs were lower

because of the existing overhead for the system and program of which this development was a part. Technology was also very mature, and the program elected to use derivatives of existing technology as possible. This was further enhanced by a flexible design, in which the front end can easily be changed. As the program manager noted however, development was not the real challenge in this program as much as manufacturing and logistics. In fact, less than a quarter of the budget is slated for development, as most of the technology is non-developmental. The system is also not highly dependent on software, and much of the functionality in this comes from look-up tables that can easily be changed.

Despite the somewhat fixed nature of the program in the hardware side, the mechanism for feedback and requirements changes exists between the tests and fielded LRIP systems and future increments. Furthermore, many of these changes can be done via software upgrades.

Program Challenges:

There were a number of issues Program D encountered in large part due to its concurrent, incremental, and rapid nature.

One of the larger challenges was with external stakeholders, who often had difficulty understanding the concurrency of the project, particularly that the increments have operational utility. Additionally, stakeholders did not clearly understand the budgeting needs of such a program, and were hesitant to fund it.

Though the program uses limited modeling and hardware prototyping, this is not used for the purpose of user feedback but rather reliability and maintenance. Again, this is in large part due

to the goal of matching current performance of the program and the short time frame in which this must be done. The biggest challenge of the program has been in testing. Despite the fact that the Program D was aware that testing would be an issue and brought in testers early, it has still been difficult. First of all, the program is limited to one test asset, which severely reduces the number of tests that can be performed in a given period. Secondly, due to the fact that the larger system of which this upgrade is a part is an ACAT I program, the testing requires a Test and Evaluation Master Plan (TEMP). At the same time, the Operational Requirements Document (ORD) was old and high-level, which would have been a challenge to derive new requirements from. As a result the program developed all new requirements. Moreover, the program office worked with Developmental and Operational Test and Evaluation (DOT&E) to develop and acquisition and test strategy that worked for both parties. Though DOT&E understands the immediacy of the program, and have agreed to the LRIP part of the program, they have not agreed beyond that, with DOT&E requesting more authority in later testing.

Summary:

Program D was once again helped significantly by its mandate and rapid schedule time. Again however, the issue of COTS use was brought up. The biggest challenge in this program appears to be testing. Despite seemingly good involvement of the testing community, the challenge for a rapid program with frequent testing here is the ability to use test assets and facilities.

The strategy here is a clear incremental process driven by schedule. There is a high level of concurrency, and increments are well planned for. This is particularly crucial in this program as the requirements are very well established, and the primary goal of the program is not to develop, but rather make producible the solution.

5.7 Evolutionary Case Study 5 – Program E

Program Background:

Program E is a series of primarily hardware upgrades to an aircraft, with some software development. It started with an existing platform with the Engineering & Manufacturing Development (EMD) phase in February 2001. Approvals for Spiral 1 and 2 in March 2002, and more recently Spirals 3 and 4 in December 2002, followed this phase. In this period, the system has had two real operational experiences. As a relative program size, the spiral 2 is on the order of 300 million dollars. More recently the system was operationally deployed.

Acquisition Strategy:

The program is using concurrent spirals to achieve capability in a short timeline. Typically spirals last on the order of several years, but do not yield fieldable results in themselves. Fielding is done through annual lots, where mature technology from each of the spirals is brought into the production lot, as seen in Figure 22.

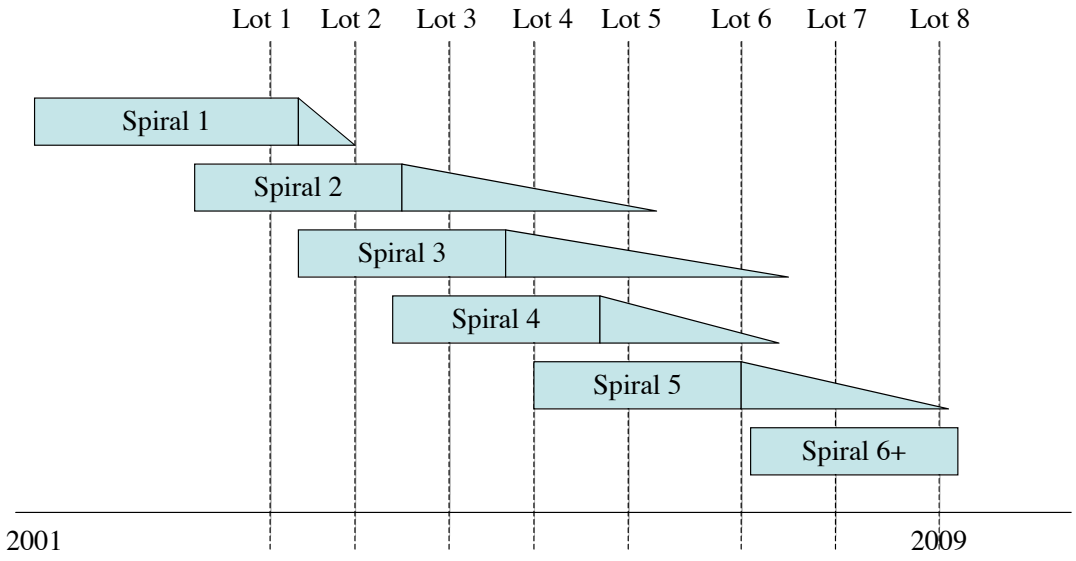


Figure 22: Program E acquisition strategy

The figure depicts both the spirals and the lots. The lots are intended to be homogeneous, and older systems will be brought up to current lot standards where possible. Accordingly, the program uses annual contracts to add more capabilities. While the platform is stable, plans are in progress to increase payload by 50 percent. Additional work in Spiral 2 is to make the system more readily producible. Future spirals will add a slew of capability improvements.

Programmatic Uncertainties:

Like many other programs, Program E was no exception to budgetary concerns, especially the ripple effects such cuts cause throughout the program. In part due to their annual contracting process, there is a significant likelihood that user priorities can shift in the course of the year, and drive the system in a different direction. Technology obsolescence was less of a concern, though hardware refreshes were expected every five years.

Key Enablers:

As Program E began it was provided with a significant chunk of the work upfront—both a platform and supporting infrastructure. It is also a priority customer to the Air Force Research Labs (AFRL). As a result, the primary purpose of the program is to use the platform to provide more capability to the warfighter. The user is very involved in the program, in large part because of the recent conflicts the system has been used in, but also because of the unique way in which this program transitioned to EMD. The user has a very clear understanding of incremental deliveries and the lots they are receiving. Like many of the spiral programs, Program E also elected to prioritize requirements, but also to have a clear requirements feedback loop through working groups.

Further feedback from experience in action as well as from testing and user experience is also incorporated into the production line, but not into current spirals. The changes can be made through subsequent spirals and in later lots. This creates a challenge for the production program however. One manager noted that there were plans for systems to be upgraded in spiral 2 and again in 5. The program has also recently had operational experience that will help shape future development work and spirals, which is not necessarily often the case for projects still in development.

Technically, the system is relatively modular within the confines of the program. In other words, because the system platform had already been established, upgrading the systems is done primarily through payload. Within the payload, the system is very modular and flexible, and incorporates a open architecture and standard interfaces. The technologies involved are also relatively mature and low risk.

The program has had some success in testing, with AFOTEC signing off on the Test and Evaluation Master Plan (TEMP) for Developmental Testing. The program managers also expressed that there were no problems with Operational Testing.

Program Challenges:

Program E expressed many of the same sentiments that other spiral programs have, again many of them a result of the concurrency aspect of acquisition. The engineering challenges with concurrent development were among these, particularly dealing with engineering multiple subsystems at the same time, and then integrating these into the lots. Another issue was the common logistics and financial issue. The managers all noted that more work was necessary in

this area, particularly in training. As a direct result of both the concurrency and the annual contracting process the program employs, the program encountered contracting problems, and needed additional resources in the contracting office. This is compounded by the fact that the program style is contract intensive, and more importantly that spirals are essentially considered large EMD contract changes. Another source of concern was the scope of the spirals. Many of those involved on the program felt that technical goals for each spiral were too large, causing schedule and cost issues. Financial issues also play a role in the program, as like many programs, operational requirements and technical changes are not possible unless more funding arrives. Of particular note is the inability to set aside money for technology refreshes or replacing subsystems that suffer from Diminishing Manufacturing Sources (DMS), as this money is quickly cut for other programs.

Like many of the programs, testing appears to be a major hurdle in the program. Most notably, the program is limited to one test asset, which is a great concern because of the potential loss of assets in testing. This problem will become more significant as the number of development spirals increases, despite attempts to test multiple spirals simultaneously. Particularly, testing the system with multiple software configurations will be a challenge—more so because software is being released approximately every two weeks.

Summary:

Program E's biggest challenges come from its concurrency, which puts a significant strain on program resources. At the same time, it gets significant benefits from being a preferred program, such as having fewer development uncertainties, as well as ease of development because the platform is established. The acquisition strategy for this program was unique among the case

studies, as it was a mixture of spirals and lots. The overlapping spirals indicates that the system is very much along the lines of the block upgrade methodology of the Air Force, and very similar to the F-16 acquisition program. Nevertheless, the use of annual lots means that the spirals are not the blocks, as much as the lots are. Essentially, one could argue that this is a variant of the concurrent technology transfer strategy put forth by Cusumano and Nobeoka in Chapter 2, where technologies from multiple spirals are transferred to annual lots.

5.8 Evolutionary Case Study 6 – Program F

Program Background:

Program F began in May 2003 as a derivative of an existing platform and is expected to last five years. The system on which this program is based was previously developed and demonstrated. An aircraft system, it is hardware-dominated, based on established, mature technologies. The program also is cost constrained from both a development and unit procurement standpoint. The program is in the mid-range of development cost compared to other programs in this thesis.

Acquisition Strategy:

As the program is just beginning at the time of this research, the intended strategy is to develop a baseline or platform, and have a delayed, but concurrent, first spiral, as shown in Figure 23. The first spiral should add new capabilities to the platform. As a result, the current strategy calls for

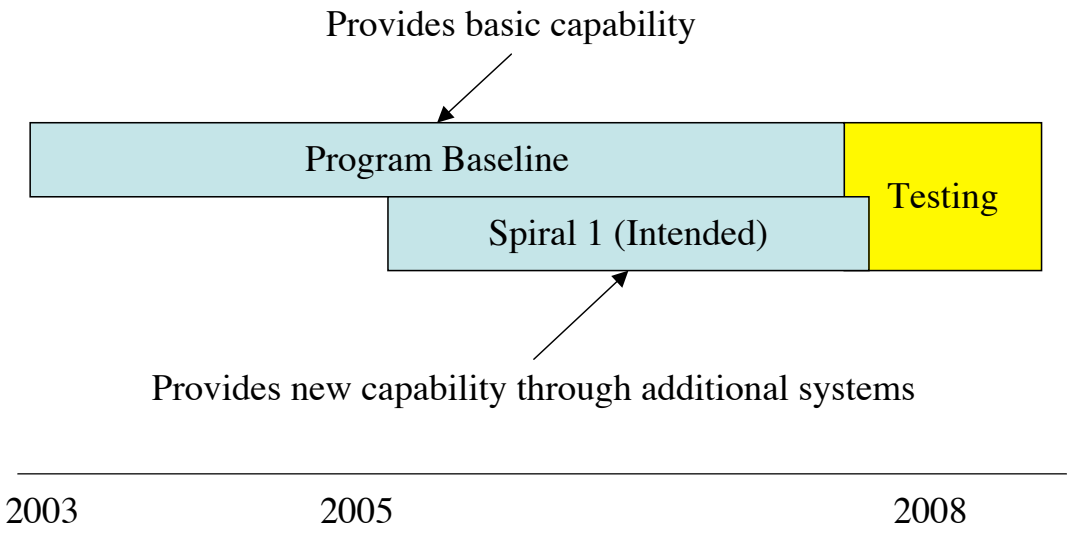


Figure 23: Program F acquisition strategy

the system to be upgradeable and allow for new capabilities to be added easily. Both the baseline program and the first spiral are funding constrained, so the program will attempt to conclude the development of these at the same time, and provide for synergy during testing.

Programmatic Uncertainties:

Like the majority of programs interviewed for this research, the program's concerns over risks and uncertainties tend towards the same areas. In Program F, the technology is mature and primarily off the shelf, but the burden of technical obsolescence is on the contractor and not the program office. In addition to this the program is seeking out emerging systems to incorporate into the development, to reduce the effects of possible challenges that cannot be foreseen. Furthermore, the program hopes to address such problems with software versions rather than hardware upgrades. In addressing operational environment changes, the program has considered many of the more likely scenarios and embraced these. Though the system is required to work with two other systems, the program has larger goals. This has the benefit not only of making the system more useful, but also creating larger demand for the system, which is beneficial from a cost perspective for both the program and contractor. The program is not particularly concerned about budgetary issues, as it is very constrained to begin with, but also enjoys favorable status with the Air Force Chief of Staff.

Key Enablers:

Though the contract for the program was awarded in May, work on the program began months before this, with meetings and teleconference between the program office, the user and the Air Force acquisition agency. These weekly meetings have continued into the program, adding the contractor after contract award. In addition to this, a large meeting with all the stakeholders, including the Air Force Operational Test and Evaluation Center (AFOTEC), Aircraft System Program Offices (SPO), and the traditional communities was performed. Working with the SPO, Air Combat Command (ACC) developed an Initial Capabilities Document (ICD) and is drafting a Capabilities Development Document (CDD) to be in place by preliminary design review. At the beginning of the program, the program office elected to perform an investigation and

solicitation from various potential contractors to find what solutions for the program needs were available and also to determine where the "knee" on the performance-cost curve was. This was done through simulation and modeling, where the contractors provided feedback based on the findings. The intent was to use these models for evaluating requirements for the program. One of the major advantages of having a concurrent first spiral concluding near the same time as the program baseline is essentially the ability to provide a single version of the system to the users to mitigate many of the traditional logistics challenges. The program also experienced a unique situation as a result of its constrained funding; it was forced to extend the schedule to accommodate the funding profile. As a result, there is not the typical challenge of scheduling as is seen in many of the other Air Force spiral development programs.

Technically, the program uses primarily off the shelf components, estimated to be about 80 percent of the system. Developmental work is primarily on the order of modifying or creating derivatives of existing systems, and the program is intentionally using low-risk technology. This is driven primarily by costs, from both a developmental and unit cost perspective. The system payload and system itself are both relatively modular, and should provide for easy upgrades in the future.

The program's test strategy has been very true to the preferred strategy of the Air Force, and the program foresees no issues with test asset and range availability. Historically, the manager noted, programs have tested large numbers of systems, but they are trying to move towards testing fewer systems more effectively, so planning for the right number of test assets has been important to the program. They have also planned for joint developmental and operational

testing, and the program manager thought that there would only be one individual operational test to determine suitability in the end. The biggest issue the program expected in testing was that costs for the range and aircraft testbeds were fluctuating and there were no clear financial figures for these.

Program Issues:

Budgeting was once again an issue in Program F, in two significant ways. First of all, as stated earlier, the program wanted to have a shorter schedule, by nearly 20 percent, but was unable to procure funding for such a program. In addition to this, the program has been unable to create funding wedges for the future spiral, since such wedges would immediately be taken away from the program as excess money. The resource issue was also an anticipated problem waiting to happen, particularly as the first spiral begins. In Program F, the weak points appear to be budgeting, cost-estimating, and user requirements planning. In fact, the program manager noted that requirements, for the spiral and in general, were likely to be a big challenge. Specifically, the manager stated that the program had to stay on top of the requirements, and prevent "what-iffing" as well as the traditional requirements creep. Due to the concurrent nature of the program, and existing plan for only one spiral, it is unlikely that there will be any feedback from the baseline program into the first spiral. The only feedback that is possible will be with improved models of the baseline program to be used in soliciting feedback for the spiral. On this issue, the program manager noted that this would be the burden of the contractor, as the Air Force program offices no longer have the capability to perform complex modeling and simulation.

Summary:

Although Program F has just begun, it has planned for its introduction into spiral development by involving the testing community early and planning test assets well in advance. Nevertheless, the program has a common resource issue, and once again is faced with the inability to create funding wedges or reserves for future use. Perhaps the most interesting aspect of this program is the fact that to develop the platform, the strategy resembles a traditional development strategy, and spirals are used to improve the capability. The program planning appears to resemble the F-16 strategy, essentially developing a baseline and following this up with improved capabilities through successive spirals.

5.9 Summary of Case Studies

Each case study provides a unique experience with Evolutionary Acquisition and spiral development, but looking at them together also provides some valuable insights into new acquisition strategies. Chapter 6 provides an analysis of these programs to determine key findings in Evolutionary Acquisition.

6 Case study results

6.1 Case study summaries

Each of the programs involved in this research uniquely identified their own processes and practices in implementing Evolutionary Acquisition. Though the programs represent a variety of product types and strategies, there are a number of common elements in each story that are valuable to other Evolutionary Acquisition programs. Following a brief review of each of the studies, these five topics will be reviewed:

- Acquisition strategy
- Perceived variation in the development environment
- Product architectures
- Program resources
- Summary of program practices

Program A, a primarily software system, embarked on a common strategy of overlapping increments, with each increment providing increased capability and functionality through a delivery mid-increment and at the end of the increment. Within each of these increments there are a number of spirals that grow the increment capability. The program also actively prioritized requirements and involved a user base that had prior experience with spiral development. The program has delivered one increment and is now working on the second.

Program B is unique among the programs for not having overlapping increments. Again a software program, it combined increments with internal spirals, though each spiral yielded a fieldable capability, in this case a software patch. The program rigorously prevented

requirements creep into its future increments without associated funding for such changes. At the same time, the program and product have successfully dealt with a major operational environment shift. Budgetary issues from a failure to agree on capabilities and funding have plagued the program. Nevertheless, the program is continuing into the second increment.

Program C is a slightly more complex system development, involving two systems (C-1 and C-2), and each has its own acquisition strategy, though they are intertwined. Essentially, work on C-1 has begun already, and related developments will feed into C-2. The C-1 program uses an overlapped increment strategy, with increment 1 focusing on both hardware and software, while increment 2 is wholly software. Again the program has emphasized prioritizing requirements, but more importantly, they have strategically worked to manage stakeholder expectations.

Program D is an upgrade program for a hardware system that involves a very well defined set of future increments, and potential spirals after the increments are complete. The program is essential, meaning that it is required for operations, so it has a significant mandate for rapid acquisition and fielding. This mandate also implies fewer challenges for the program, as the stakeholders understand the program necessity and are willing to put in effort as required. Fundamentally, the program is spending little on development, and much more on logistics and testing. It is a very concurrent program, and this has been the biggest challenge for them.

Program E is primarily hardware development for an existing platform. Several of the spirals are currently under way, with each spiral focusing on a particular capability. Running orthogonally to these spirals are production lots, which cull all the technologies ready from each

of the spirals into annual production lots. Though the program is a preferred program and enjoys popular status, it suffers from resource challenges associated with such a large endeavor. The biggest advantage in this program, though it has been somewhat difficult, has been the ability to put the system in real action, leading to a better understanding of not only performance, but also operational utility.

Program F is a new hardware platform development starting with a program baseline development and mid-baseline planning to begin a first spiral to add capabilities to the platform. This program also has a favorable status and hence has fewer concerns with some of the traditional budget woes. Interestingly, the program was forced to extend the contract out beyond what the contractor had said was required in order to address the budget profile. The program did very extensive work before contract award, spending significant effort in involving stakeholders as well as looking at various options to satisfy the user needs. The program is currently working on the baseline program and planning for the first spiral.

6.2 Acquiring new products

Though the case studies showed several strategies for developing new capabilities, there were many similarities. The acquisition strategies are not necessarily the same as development strategies, and this research is far more focused on supporting Air Force acquisitions, with the understanding that these processes are interactive. Figure 24 shows a notional idea of the interactions that lead to acquisition strategy. The policy, product, and acquisition strategy are all

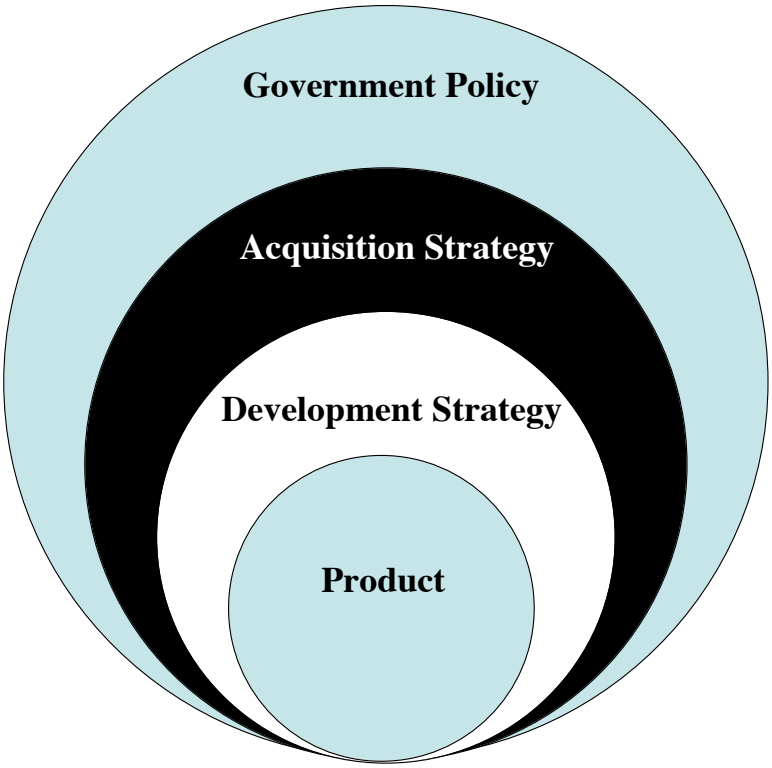


Figure 24: Notional view of process influences

within the control of the government for the most part, while development strategy is typically determined by the contractor, but clearly influenced by these other factors.

In the case studies, there were a number of acquisition strategies, but several common themes running throughout. The most evident of these is the concurrency of the projects. Looking at the programs shown in Table 1, several have an extremely high level of concurrency.

Table 1: Comparison of Case Study Programs

Program Name	Program Size	Level of Hardware	Level of Concurrency	Number of planned Spirals/Increments
A	Small	Very Low	Low	20+
B	Small	Very Low	Very Low	15
C	Medium	Low	Medium	4/8
D	Large	High	High	3
E	Large	High	High	6
F	Medium	High	High	1

More importantly, there is a noticeable trend that larger programs and/or programs with more hardware have higher levels of concurrency. A closer look at these programs reveals that they are very similar to the incremental process described in Chapter 3. This process involves a significant portion of the design being planned for up front, while deliveries of the product come in increments. In other words, these programs had determined the essential content of each of these increments in advance, although this was less true for increments well into the future.

A crucial aspect of the ability to iterate, to evolve, and to experiment with new capabilities is the ease with which changes can be performed. That is to say that the difficulty of creating derivatives and incremental families of products is a function of the ability to and ease of change. In hardware programs, where the cost of experimenting is typically higher and requires more work, it is less likely that programs have the budget and schedule freedom to experiment. Furthermore, the potential cost risk due to failure is higher as well. Conversely, in software programs, or in low-cost programs, where the cost of iteration and experimentation is low, there seems to be less concurrency and up-front planning.

Of note is the number of total iterations among programs. Counting the spirals as iterations in Programs A and B, it is evident from Figure 25 that the lower-cost, easier-to-change programs show a significantly higher level of iteration when compared to the higher-cost, hardware-intensive programs. One director interviewed noted that a significant problem with hardware

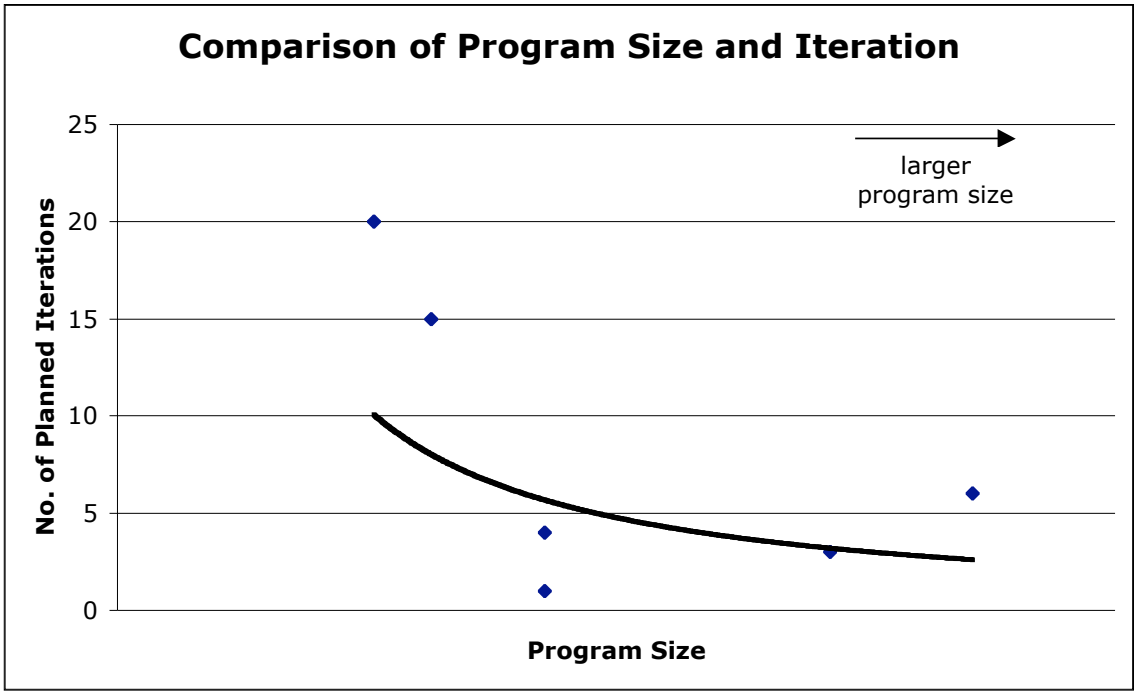


Figure 25: Program size vs. level of iteration

programs was the challenge in quickly acquiring new hardware due to lead times from contractors and manufacturers.

Larger programs also typically did not have spirals within the increments, as did the smaller, less hardware-based programs. In fact, the primarily software programs had clear spirals within the increments, and made efforts to field these spirals. These programs did so by requesting such deliveries from their contractors. Only Program E was fielding technology from spirals that were yet unfinished, though this was done primarily through the use of production lots. While

some programs used the opportunity of fielding to incorporate new feedback and experience into subsequent spirals or increments, there was no clear consensus about this. One of the software programs, despite having little concurrency, refused to accept changes in plans for future increments without the associated cost of changes. Indeed, several programs cited this as a significant challenge. Despite the ability to incorporate new feedback, funds were not always available to do so. In addition, as many programs were concurrent, few were able to immediately incorporate feedback into the subsequent increment or spiral but many were able to place changes in future increments. At the same time, all programs were adamant about fixing mission critical changes immediately after fielding.

Actual processes – theoretical processes

Though program strategies varied somewhat, it is important to realize that none of the programs fell cleanly into one of the theoretical processes described in Chapter 3. Though each of the programs was compared to the theoretical strategies in the previous chapter, looking at them together shows several commonalities and aspects of existing strategies.

First of all, none of the programs is performing a spiral development in the pure sense of the work by Boehm, Highsmith, and others. Spiral, in the traditional sense, applies to growing performance while reducing risk, but not necessarily doing so by providing deliverable capabilities at each spiral. In fact, the formal spiral model describes a process by which increasingly defined prototypes are delivered to the user until a fieldable, final capability is reached. The goal in this strategy is not to reduce development cycle time, but rather to address uncertainties—particularly user uncertainties—through iteration and flexibility. At the same time however, one could argue that the Air Force strategies leave some flexibility in future spirals, but as a result of the planning process, programs must commit to follow-on spirals before the

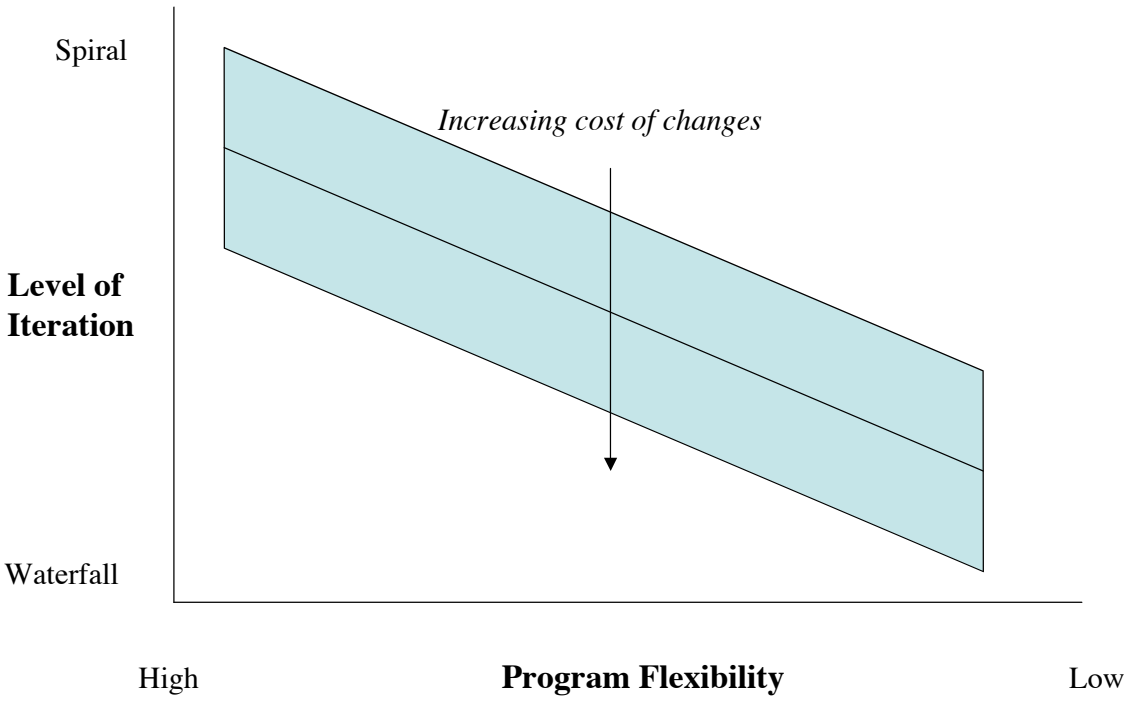


Figure 26: Notional diagram of program flexibility and iteration

previous one is complete. Notionally one might compare the processes on a scale of flexibility versus iteration, notionally shown in Figure 26. This figure shows a general relationship between the program flexibility and the cost of changes as compared to the level of iteration expected. Software programs tend to be easy and inexpensive to change, while hardware programs are typically more complex and more expensive to change. Historical examples of pure spiral development are inevitably software or commercial computing hardware. The result is that Air Force programs represent a hybrid between spiral development and incremental development. The exception is Program D, where the program has essentially committed to three increments up front that are immediately necessary and also well defined. In fact, this wholly makes sense for this program, as there are no user uncertainties, and unnecessarily expanding the process to a spiral makes little sense.

Interestingly, almost all programs are using a Design-to-Schedule or Design-to-Budget strategy within each of the increments. As programs are typically cost and schedule constrained, this is very logical. In the case of the programs examined here, these tended to be more Design-to-Budget oriented. Figure 27 describes such a process. Note that in this process, the requirements

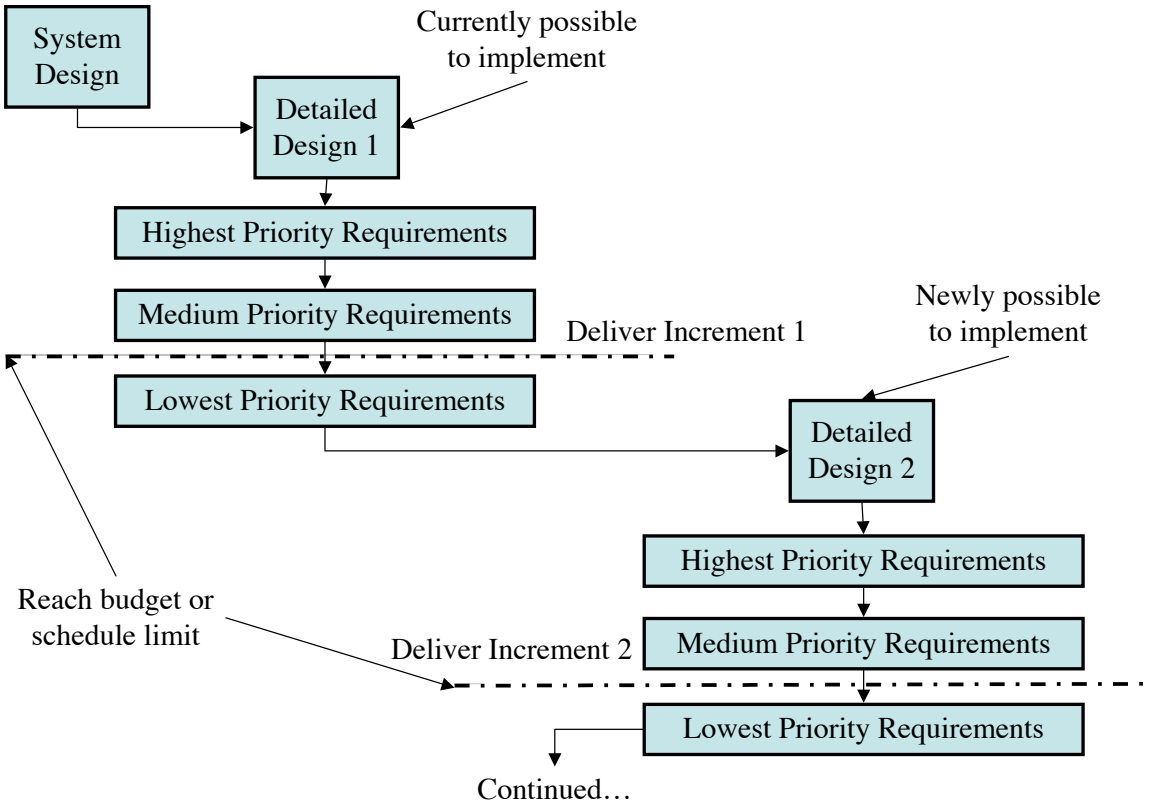


Figure 27: Design to budget/schedule in an incremental development

that cannot be completed in each increment are moved to later increments. Also note that each increment only focuses on what is possible to implement in that period—i.e. what is mature and ready.

While the programs all fall into the category of Evolutionary Acquisition, there is obviously some discrepancy between programs as to what that means, and how best to incorporate the new strategies into the traditional acquisition environment. For example, Program B has notable

iteration and an opportunity for immediate feedback as increments are not overlapped, while Program F currently has no way of implementing recommendations based on experience.

6.3 Uncertainties in the product development environment

It is expected in the course of an acquisition that things will invariably change. These changes can come in a number of ways—they can be changes in the funding profile, changes in the user requirements or needs, external advances in new technologies, obsolescence of technologies used in the program, even changes in they way the product is anticipated to be used. As a result, throughout the product development process, program mangers are expected to manage these uncertainties and the associated risks. As stated earlier, it was hypothesized that strategies would vary based on the types of uncertainties the program expected.

Within the programs interviewed however, there is no clear correlation between these concerns and acquisition strategy. Table 2 shows the programs and their manager's concerns.

Table 2: Uncertainties by program

Program Name	Uncertainties
A	Operational Requirements, Budget
B	Operational Requirements, Technology Obsolescence, Budget, Operational Environment
C	Operational Requirements, Technology Obsolescence, Operational Environment
D	Operational Requirements, Technology Obsolescence
E	Operational Requirements, Technology Obsolescence, Budget, Operational Environment
F	Budget, Operational Environment

Each of the programs dealt with these uncertainties in several ways. Of interest is the fact that almost all the programs had operational requirements and environment concerns, and the

majority had concerns on technological obsolescence, in large part due to the fact that they were evolutionary programs.

Operational requirements

As operational requirements uncertainty was a large concern of programs, it is interesting to look at this and understand both its reasons and solutions. Requirements creep is always a concern in programs, and its effect seems to grow with the length of programs. This is likely the result of a natural course of events: 1) User's needs and desires change with time and 2) User's needs and desires change as a result of knowledge learned through the course of the program, either through models, prototypes, or fielded capabilities. The effect is more pronounced with Evolutionary Acquisition, as there are discrete intervals for inputting new requirements before spirals or increments begin. Nevertheless, due to the concurrency of many of these programs, the increments and spirals are relatively well planned for in advance, and there is little desire on the part of the program or contractor to change requirements. This is particularly key when users request changes without the additional funding or possible alternative areas to be cut. The challenge in these programs is between engineering desires to establish requirements and program desires to remain adaptive. Prior research by Dare (2003) indicates that one way to deal with the uncertainty in the acquisition environment is the ability to adapt a system while is being designed. Specifically, Dare cites the type of collaboration between stakeholders as the primary determinate of system adaptability. His work shows that programs with a high level of collaboration are more likely to be adaptive.

As a result, programs have developed a couple of key strategies for dealing with this. The first of these is to heavily involve users in the early planning stages to minimize requirements

changes due to misunderstandings or poor analysis of user needs. The second of these is to prioritize the requirements with the user, so that the most immediate and important requirements are satisfied as early as possible, and lesser requirements are left to future spirals or increments. Both of these strategies tend towards fixing requirements, but this seems to contradict spiral development's goal of maximum flexibility. In fact the two can coexist, but the key is to hold only those requirements that are certain, and keep flexibility in the less certain requirements. In other words, a mix of freezing known, static requirements and iterating on unknown requirements is a valid strategy for acquisition. Nevertheless, this provides a challenge in itself, as users tend to over-prioritize many requirements, which is in turn a result of their expectations in Evolutionary Acquisition programs. That is to say that users often do not understand the principles of Evolutionary Acquisition, to provide incremental capability over time, or they are hesitant to believe that the program will fulfill their end needs, often because they anticipate program development or funding cancellation before completion. The response from a couple of the program managers has been to try to further educate the user and to manage expectations. The alternate approach has been to firmly deny requirement changes unless they are accompanied by additional funding.

Technology obsolescence

Technical advances and technology obsolescence has proved to be a significant part of any development task, particularly in the aerospace industry, where typical cycle times are on the order of 10 years. Surprisingly, this was a concern in software programs as well. The fundamental problem in this case is that technologies underlying the performance of the product being developed often evolve more quickly than the product does. As a result, one of several things can happen:

1) The program can continue to use obsolete technology, and attempt to create a significant accumulation of the parts necessary through the product lifetime
2) The program redesigns the product to incorporate new technologies
3) The program suffers from a challenge of Diminishing Manufacturing Sources (DMS), such that there will be a need for a mid-life upgrade in the product lifetime.

Nonetheless, any of these is undesirable, as all require additional work and/or money. Programs studied here attempted to deal with this in mostly the same way.

The general approach to this, particularly for programs that require hardware has been to plan for technology refreshes throughout the product development lifecycle. That is to say that they have planned for upgrading the technology in future spirals. The primary methodology of doing this has been through the use of Commercial-Off-The-Shelf (COTS) products, which often reduce development costs, and to follow commercial interface standards. This is possible in some cases, but in others, due to operational environments and government policies, this is not feasible. Furthermore, COTS products also result in a number of challenges in themselves. In fact, COTS products themselves tend to obsolesce more rapidly than traditional Military Standard (MILSTD) parts. On the other hand, COTS products have the reputation of being easier to upgrade, though this is not borne out by the experiences of some program managers. In particular, these program managers note that in software, using COTS is a huge risk, as new software is not always backwards compatible, and it is not always possible to use software from different vendors. Logistically speaking, the seemingly short technology refresh cycles pose a challenge to logisticians, as they must prepare to upgrade all the existing systems or deal with multiple configurations of the same system.

Budget

Programs A, B, and E both had some concerns over budgeting cuts in the program future, but clearly this is beyond the control of these programs. At the same time, Program B's manager noted that there was a clear inability on the part of the user to understand the cost-capability tradeoff. In particular, the issue was that the user did not have the engineering savvy to understand which capabilities could be traded and which were technically necessary to the product. This can be attributed in part to the inability of the program to adequately manage user expectations. That is not to say that some programs had no budgetary concerns. All of the program managers involved had concerns with budget cuts, in large part due to historical experience or precedence. Program A noted that it dealt with this challenge through the use of its prioritized requirements, which provides flexibility to deal with the cuts by understanding which requirements are more flexible. In essence, as program managers are no longer able to keep budget reserves to deal with unforeseen circumstances, they use requirements instead, sacrificing those less important requirements if funding is cut or if development changes are necessary.

The larger problem in Programs E and F budgeting had more to do with the ability to hold their budget under their control. In particular, these programs were unable to establish their own management reserve or funding wedge for future needs, such as dealing with unforeseen technological advances or testing issues, and so forth. In particular, managers noted that other programs would appropriate any 'excess' or unused money.

Operational environments

Operational environment changes were experienced or expected by several of the programs throughout the product development lifecycle. The ability to control this is not possible to the

programs, but strategies varied among programs as to how to deal with this. Program B experienced a significant change as a new group of users arose during the development process requiring some additional development work. As the program was software based, this appeared not to be a significant challenge. Program D arose as a direct result of an operational environment, but does not expect additional changes in the product lifetime. In programs C, E, and F, all of which have significant hardware, there are several opportunities for taking the product into a different operational environment. Program E is aware of such possibilities, but in

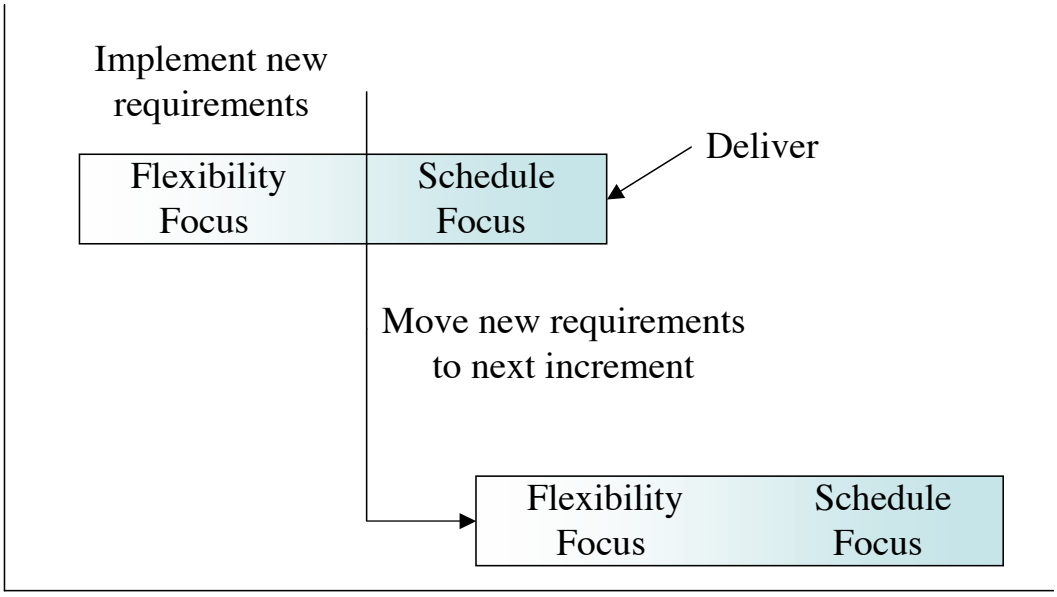


Figure 28: Implementing change in programs

their case this is likely to be handled by the contractor. Program C is aware of the potential for these changes and in discussions with the potential new users. Program F, as it is early in development has planned to incorporate new users into its product capabilities. In many cases the shift to a new operational environment is the result of user realization that the program has potential to be used in a different way. This is very beneficial from a user point of view. If programs are made aware of this in advance, this is not so detrimental as changes early on are typically less costly. However, new operational changes late in development could potentially

be handled more easily as new increments focused on creating successive variants of the nearly completed system, as demonstrated in Figure 28. In this way knowledge can be shared from the old program to the new program without significantly impacting the older development.

Summary of uncertainties in the product development environment

All of the programs experienced some level of uncertainty in their development process, but there was no obvious single uncertainty that dominated all the programs. At the same time, programs that experienced certain uncertainties, such as changes in operational requirements, dealt with these in much the same way. Significant operational requirement changes were either ignored unless adequate funding was made available, or alternatively the stakeholders were brought in to development teams to better manage expectations. Programs that used prototyping and/or modeling were also able to use these to provide the user with a better understanding of the system as well as gain valuable experiential insight from the user. In programs facing technology obsolescence, the pervasive strategy has been to plan for technology refreshes in the future spirals. While all programs were concerned about budgeting cuts (with the notable exception of those that were already very constrained), the larger problem was the program's inability to carve out funding wedges for its own unanticipated needs. Lastly, while most operational environment changes are not controllable, programs dealt with this by engaging potential new operators early on.

6.4 Product technologies

In chapter 2, there was a significant discussion on system modularity and the use of open architectures. While theoretically, this is very valuable, implementing this in programs with real products is not always easily done. The Air Force does encourage the use of open architecture, modular systems, but this is a challenge that many programs face. Many times, these choices are

out of the control of the program office, and are often based on performance needs such as in the case of the F-22.

Looking at the various programs, there are three aspects of the system architecture that are valuable to further investigate. As stated earlier, those programs where it was easy to experiment and develop new capabilities gravitated toward more iteration. Figure 29 provides a different perspective, looking at three key factors in the design of the product.

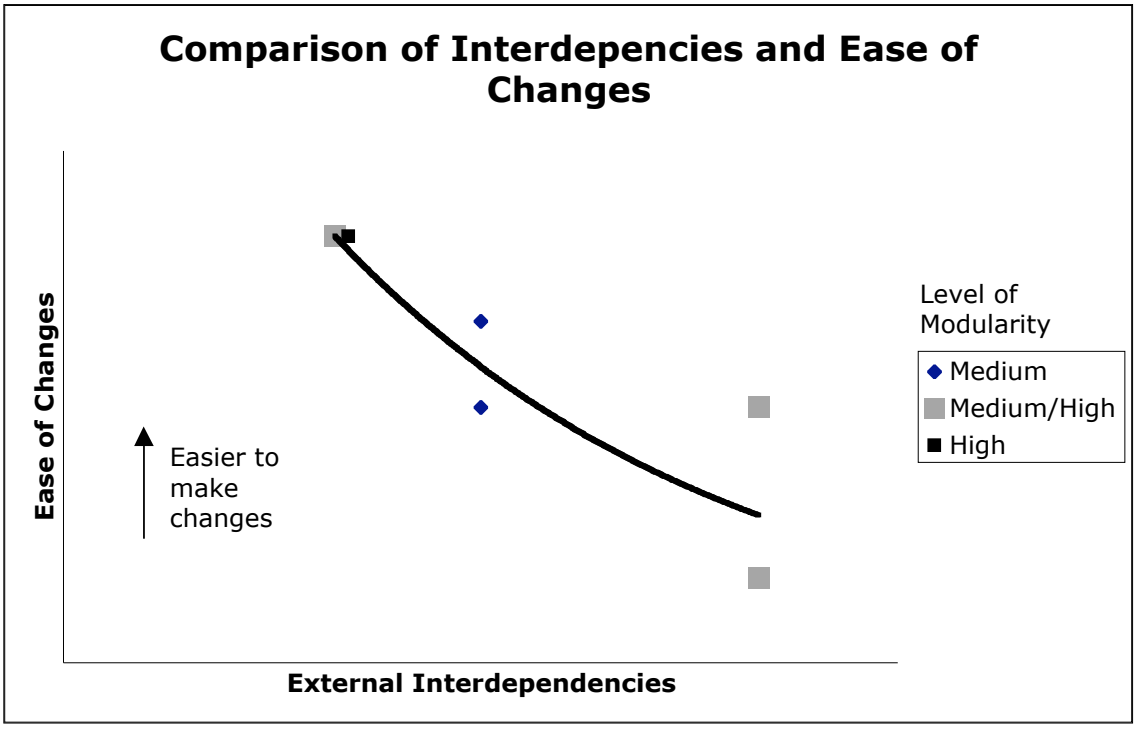


Figure 29: Program interdependencies vs. ease of system change

The figure shows distinctly that the level of modularity in the system is far less crucial than the level of interdependencies in terms of ease of changes. That is not to say that modularity has no effect, but external interdependencies dominate the ability of the program to make changes. This implies that programs must focus on the level of interdependency more than on the system architecture in order to evolve. At the same time this indicates that in a tumultuous environment,

evolving systems at the rate of the environment will be a challenge. The fundamental goal of the program must then be to match this external evolution and the evolution of the system.

The level of modularity is a valuable indicator of the way the system interacts internally—whether components internal to the system can be changed with ease. The level of interoperability is an indicator of how the product interacts with other systems, and whether the product must be designed around existing infrastructure and systems. The ease of changes is an overall metric of the program's ability to change without negative consequence either in terms of sacrificing program performance or reducing the ability to operate with external systems.

It is interesting to note that the software programs once again had a high level of internal modularity, while hardware programs had some flexibility, but there tended to be greater interdependency in the subsystems. This is somewhat expected, as software tends to rely on libraries or modules to execute code, the key being that the data passed between modules must remain the same. Conversely, with hardware systems, there tend to be centralized systems handing multiple tasks, in addition to a greater potential for interference of one system with another. In Program E, the modularity tended to be within the payload, but not necessarily in the other systems. Program D's modularity within the main development task was good, as their subsystem broke up into somewhat clean functions. The same was true for Program A, as the capabilities separated nicely into individual functions that could be addressed by individual modules.

A bigger driver of the ability to change the baseline program was the level of interoperability. Only one program did not have significant requirements on interoperability, Program A, as this program was essentially replacing the old system, and the functions were clearly understood, so that the data being passed along from the system to other systems was well defined. Programs with hardware, such as D, E, and F tended towards greater interoperability, as they often had to interact with infrastructure and systems that are themselves never completely constant. Program B is the notable exception to the primarily software systems, as the program had to operate with a larger software system, and this system itself was in a constant state of flux. As a result, ensuring interoperability with this system required Program B to both adapt to the external system as well as to make sure that its actions did not detrimentally affect the system.

As a result, the final indicator of ease of change is a mixture of both the level of external interdependencies and the level of modularity and the program manager's perceived idea of how easily the product can be changed. Programs A and C, were relatively easy to change, but Program B, for the reasons stated above was driven heavily by external circumstance. Programs D and E were forced to some interoperability, but because the rate of change of these external systems being relatively slow compared to the development cycle, this was not a major hindrance. This was particularly the case for Program D, where the subsystem being developed was to be used with an established system. Program F has the virtue of being very easy to change, but this was primarily because the system is as yet not strongly defined. In the end however, it is required to be interoperable with several relatively established systems.

As a result of the system of systems approach in modern engineering, and particularly because the Air Force has a large base infrastructure, it is somewhat expected that interoperability will always be a concern with hardware systems. With software systems, this will vary much more with the type of system involved, with Program B being the notable exception because it operates in a network. In fact, one source early in the research pointed out that networked software systems, particularly secure networks, would require significant work to change and verify as these information systems are considered critical to defense. One staff member in Program C pointed out that another issue with both internal and external modularity had to do with the way data was passed between systems, and that it would be virtually impossible to have a true open architecture system until the Air Force would purchase data rights to the systems they procured, allowing them to pass information between systems. In this regard, these systems are analogous to computing today, where passing information between proprietary systems such as Microsoft Office and other productivity suites is difficult, but systems with open or free data rights, such as HTML code, can be passed between a number of programs and systems. Once again, the level of external interdependencies is not always easy to change from the program perspective. Nevertheless, the biggest problems evidenced were when the external systems were evolving faster than the system itself. One way to mitigate this is to match the program cycle time to the external cycle time, by appropriately selecting content for each iteration. Content selection was in fact a significant issue noted by many programs, with the notable exception of Program D, which had planned the increments to match its well-known environment evolution.

6.5 Program resources

One of the major challenges of spiral development according to its many proponents is the level of management required to successfully execute a spiral program. In casual discussions before

embarking on the case studies, a couple of people with spiral program experience noted that the program resources were a fundamental challenge. As a result, programs were asked about where they had inadequate resources to handle the additional requirements, if any, of spiral development.

Programs suffered from a variety of challenges, as shown in Table 3.

Table 3: Program resources

	Resources Necessary	
Program Name	More resources	Fewer resources
A	Planning, User	
B	Cost estimation, Planning, Testing	
C	Contracting, Logistics, Planning Testing, User	
D	Logistics, Testing, User	Contractor
E	Contracting, Engineering, Logistics, Planning, Testing	
F	User	

Most programs indicated that spiral development required more resources and buy-in from various stakeholders. It was more interesting however, that these resources were very similar for most of the programs.

Contracting proved an issue for both Programs C and E, but especially the latter. In the case of Program C, the program spent a lot of time working on contracts, and as one manager stated, they needed a contracts officer who was more flexible. This is likely due to the issues with the inherent flexibility of spiral systems, but traditional contracting needs have not kept up to this change. Program E had a similar problem exacerbated by the annual contracting process they

used for lots. In addition to this, because each spiral was considered a large EMD contract change, this created an additional burden on the program.

Cost estimation was a problem with Program B, as it experienced a challenge going from approved spending based on initially estimated costs to a newly estimated cost as the start of a spiral began. In other words, the program had difficulty accurately estimating costs early on and was able to provide a better estimate only as more details became available. Unfortunately, this new cost estimate was higher than they had been approved for, causing some issues. Conversely, Program A's manager stated that as a result of their experience in the first spiral, they were much more able to provide accurate estimates for the second spiral.

Engineering proved to be a challenge for Program E as they were faced with the daunting task of integrating across multiple spirals. Other programs also briefly mentioned that it was difficult to have engineering work on prior spirals once a new spiral had begun, as they had been shifted. This was typically during the phase that the prior spiral was completing testing, and some engineering changes were necessary. This challenge comes as a result of the concurrent development process, where corrections to prior spirals must be made at the same time that new spirals are being developed.

A number of programs experienced difficulties with logistics, particularly those with some level of hardware involved. Program C's problem was somewhat unique in that their use of COTS contributed to logistics problems, especially in computer hardware, where users would often change some components without approval. For Program D, the problem is going through a

series of increments, where some systems are upgraded in a given increment, and others are brought up to the same capability in later increments. Program E however, encountered a larger logistics problem as a result of its lot development strategy, which led to multiple fielded configurations. Furthermore, the challenge also was in training users to operate the system. Certainly as multiple configurations of systems are released, if no efforts are made to bring older systems up to new standards, then it is likely that operations and sustainment costs will increase. This is a potential challenge in evolutionary programs.

Planning was an oft-cited challenge for many of the programs. This was particularly noticeable in programs with a high level of concurrency between ongoing development and planning for subsequent increments or spirals. The noticeable exception to this was Program D. In their case, they elected to plan for all the increments up front, and deal only with development, testing, and delivery in each of the increments. That is to say that the contents of each increment were determined at the program outset. Program F has not yet encountered such a challenge, but the program manager did expect that there might be some resource limitations as they began the first spiral.

Though many programs were acutely aware of government policies on testing, and involved the Air Force testing institutions in their planning and development, this did not cure the problem of testing asset availability. Test assets were often the biggest challenge in hardware programs, but regardless of the program type, the ability to rapidly test systems was missing from most programs. Programs often had many of their own test facilities, and several programs admitted to hiring additional testing staff. The challenge was particularly the result of the number of test

cycles a program had to go through. Because of the number of deliverables and current regulation, programs had to test at each increment, for the most part as if each increment represented a new product. Program manager B noted that a lot of the testing was repetitive, and another program manager suggested that a regressive testing of only systems that have changed might be more effective. As programs iterate, there are more opportunities for testing, and it is likely that given the current practice of testing every fielded capability as if it is new will increase testing costs compared to traditionally developed systems.

The majority of programs spoke highly of their involvement of the user community in the development and decision making process, but at the same time, these programs also demanded quite a bit from their users. In particular, because of the programs' decision to deliver increments rather than a single, final product, the requirements process leading to increment definitions is more intensive. A number of programs opted to prioritize requirements and execute top requirements early on. In addition to this, programs cited managing user expectations as very valuable, and user involvement with other stakeholders was noted as one way of doing this.

Though the majority of programs noted resource shortfalls—which is to be expected as these are more obvious, and few programs have an abundance of resources—Program D stated that they had less resource requirements on their contractor's part, as a direct result of the concurrency of the project. The reason for this, the program manager stated, was because with planning the program early on there was no period of decreased activity for the contractor, and therefore no need for a "standing army."

Programs on the whole seemed to have significant resource burdens, most of which were attributable to the concurrency of the projects. At the same time, the incremental nature of the process seemed to also add some resource challenges, particularly in managing user expectations and understanding user needs.

6.6 Summary of program practices

There were a number of valuable practices in the case studies, but several that helped to deal especially with the challenges in an Evolutionary Acquisition environment. It is useful to understand what practices help deal with both the uncertainties in the product development environment as well as with Evolutionary Acquisition.

Stakeholder involvement

One of the most powerful practices that the programs have implemented is properly involving key stakeholders early on in the development process. This typically includes the user, test, and logistics, particularly in evolutionary systems, where these areas typically provide the most challenges. That is to say that in highly iterative programs, the challenges encountered once in a traditional waterfall process are encountered repeatedly. Accordingly, planning for these significant challenges with the stakeholders becomes all the more important.

Programs successfully did this in two ways. The first was very simply to use prototypes or interim systems to engage the stakeholders and more importantly to gather feedback on how to continue the program to best suit the user needs. This was done by demonstrating capability, and not necessarily the actual solution. The second strategy programs employed was to involve stakeholders in the requirements prioritization process. This allows the program to gain a better

understanding of the user needs as well as provide valuable information in selecting a development plan.

Managing stakeholder expectations

The programs that have arguably been the most successful have also done a very good job of managing their various stakeholders' expectations. In particular, the most important group has been the user. Specifically, programs must be able to set realistic expectations for each increment, both from a technical perspective as well as satisfying the user needs. As one program manager stated, the first increment might be the most crucial, as it must provide adequate capability, sustain legacy and new equipment, and most importantly satisfy the user so that they will accept the product and continue with an Evolutionary Acquisition strategy. This is in part a result of historical experience with waterfall programs delivering full capabilities as the first or only increment, as well as user fears that programs will be cancelled before the user gets all the capabilities the user wants. This is a challenge for some programs, as providing a better capability than the previous system on the first increment is especially challenging, and the user must be made aware that the first increment is one of many to provide significantly better performance to the user in the end.

One successful way of doing this was the use of change agents in the user community—people who had a good understanding of evolutionary programs, and explained it to others within the community—an inside job, so to speak. Furthermore, keeping the user up-to-date with the program and keeping them engaged, often through the use of system representations or demonstrators, prevented any major misunderstandings.

Planning for test

Testing has always been a challenge within the acquisition community, but these difficulties are magnified by the iterative nature of Evolutionary Acquisition. Testing must occur more frequently in incremental programs, and as a result, more resources must be applied to this process in addition to more efficiently using resources by bringing in the test community.

Programs typically dealt with this typically by establishing dedicated testing facilities and assets, which seemed to ease many of their woes. A few programs even challenged the testing process itself and were able to see testing schedule reductions of nearly 50 percent. Programs outside of these case studies have often struggled with testing schedules as well, but have had good success in improving this process through Lean and a number of other strategies. A significant issue with current acquisition is that the established regulations and protocols do not necessarily address highly iterative programs going through frequent testing. While one program manager suggested regression testing, this matter of how to reduce test times is very much a policy concern.

Platform development

While few of the software programs indicated that they were performing platform development, or were using existing platforms, in two of the more successful cases the programs were using established infrastructure, which could be called a platform so to speak. Essentially, the inputs and outputs of these systems were already well defined, simplifying the development task. All of the hardware programs had or will have a platform for continued Evolutionary Acquisition. Program D was a subsystem that essentially was a derivative of an existing system, while Program E has an existing platform and is working primarily on payloads. This is particularly

important to realize, as evolution can occur on a much more rapid timescale once the platform is established, as shown in the F-16 example.

This poses a difficult challenge for Evolutionary Acquisition programs—development of platforms that meet or exceed the previous generation of systems typically does not offer the reduced cycle times that the Air Force is seeking. One solution to this in the future is to primarily focus on systems that can be evolved or spiraled for long periods of time. This works well in systems where there are few disruptive technologies or major shifts in operational requirements and environments. Platform development is essentially necessary only when the evolution of existing systems cannot provide a new, necessary capability.

Minimizing interdependencies

As shown earlier, in both Chapter 2 and in this chapter, modularity can be very valuable in allowing for program flexibility, in both addressing development challenges as well as incorporating new capabilities. Beyond this, it is imperative to try to minimize interdependencies with external systems where possible. Through doing so, programs can evolve at their natural frequency. This is made more important by programs like Program E, where spirals are focused on specific capabilities, and these must be integrated together.

While it is easy to say that modularity is essential to rapidly evolving systems, implementing this is not as clear. It is interesting to note that evolvable programs such as the F-16 arose at a time of military standards and not necessarily off-the-shelf technologies. What is truly essential for the Air Force (in some cases) is the ability to rapidly insert new capabilities into existing systems. Controlling the interfaces and establishing standards best serve this purpose. In other

words, as one program manager pointed out, despite the use of commercial technologies, the bigger problem was in data rights—meaning that the program could not transfer data back and forth between various systems without the data rights. In other words, in the software arena, data transfer, i.e. inputs and outputs, are the essential interfaces between modules and subsystems. As many commercial programs use proprietary data formats, different modules of software are not necessarily compatible, and as this interface is not controlled by the Air Force, it becomes a challenge to integrate new modules or capabilities. Using established standards for interfaces would allow the Air Force to quickly integrate new systems. Furthermore, it is essential to note that when interdependencies do exist that the systems involved must evolve at a similar rate, or else significant work is necessary to keep up with the faster moving system. This was a significant challenge for Program B.

Requirement prioritization

If managing user expectation is a fundamental part of Evolutionary Acquisition, then understanding the user needs is as well. As programs are often cost and schedule constrained, it becomes crucial to understand what is possible in a given increment. In much the same way that the Design-to-Budget or Design-to-Schedule strategies from Chapter 2 work, programs can successfully use this to understand what works in a single increment. Doing so means prioritizing the development tasks, which in turn means understanding what is most immediately important to the user. Furthermore, prioritization allows for flexibility in budgeting, as programs can continue to execute development tasks until the new budget limits are reached.

Using a Design-to-Budget strategy within a given increment is a logical way of dealing with many of the funding uncertainties in the current climate. At the same time it provides valuable

insight into user needs. If programs are truly to execute pure spiral development, it is necessary then to keep only those requirements that are absolutely certain and adapt the rest through the development process.

Experiencing capabilities

In flexible programs such as those using evolutionary processes, it becomes increasingly important to solicit feedback from the user in terms of operational experience and capabilities. There are two ways to do this—actual experience in the field, and prototyping or modeling. While little is a substitute for field experience, prototyping and modeling in early stages can provide the user with an idea of what can be expected in the field. The result is that the user can in turn supply better feedback to the acquisition community, and influence the development course the program takes.

In Robert Dare's work on stakeholder collaboration, he found that the use of system representations provided not only a means of engaging the user, but also a way to provide the user with experience so the user could in turn provide better feedback to the developer and program. This is a necessary part of Evolutionary Acquisition, as users must be given the opportunity to provide feedback based on use and evaluation to shape future development.

Results summary

Though the information was not available to answer several of the questions brought up in the literature review, a number of key findings were made that will provide for a better understanding of Evolutionary Acquisition and implementation. In addition to placing the various development strategies into the context of the product development processes established in Chapter 2, a number of interesting findings about the ability of programs to iterate were

established. Furthermore an overview of the uncertainties these programs encountered in the development process uncovered a number of key strategies for dealing with the uncertainty. This was followed by a length discussion of resource challenges in implementing Evolutionary Acquisition. Finally, the chapter concluded with a highlight of the best practices established by the various case studies.

7 Conclusions and recommendations

Though Chapter 6 discussed a number of findings of this research, it did not answer some of the hypotheses and questions put forth at the outset of this work. While a number of these questions cannot be answered as a result of the failure to capture the meta-data necessary for analysis, several of the key questions regarding Evolutionary Acquisition and implementation can be answered. Again the author would like to note that the more significant results of the related research should be released in a paper sometime after this thesis publication.

In the introduction, three key questions were put forth based on the needs of program managers in both the Air Force and in other product development projects. These were:

1. What knowledge is necessary to adequately select a product development strategy?
2. When does Evolutionary Acquisition make sense?
3. How can the acquisition community best implement these various processes, particularly Evolutionary Acquisition?

This chapter will address each of these questions in turn, and conclude with a section on further research opportunities and a brief summary of the research conducted.

7.1 Selecting a product development process

Selecting a product development process is not necessarily an easy choice. While on the one hand there are a number of strategies available to program managers, there are a number of factors that must be considered in choosing a process—a daunting task in itself, let alone the actual development. Chapter 3 outlined a number of these strategies based on historical literature and studies. While this research was not able to span the breadth of these processes, the literature review indicates a number of key findings relating process to product.

The first product attribute that should be considered is the technical risk. In performance-centric products, particularly in the Air Force where performance means a generation of improvement over existing systems, the largest risk is typically technical. Technical risks, however, do not coexist easily with flexible, adaptive systems. The typical response to dealing with these risks is the traditional systems engineering approach of managing technical risk. This involves a series of development milestones and rigid reviews to ensure that the product passes each stage with no identifiable problems. This is along the lines of the waterfall process, and some of its variants. While a none of the programs studied in this research had significant technical risks—most were using mature, developed technologies—it is important to note that these programs accordingly followed a much more incremental or spiral approach to development. In fact, the one program with the most significant amount of development or technical maturation required, Program F, also had a long first increment that resembles a waterfall process.

The second attribute that is essential is the user uncertainty. This will vary from program to program, but will essentially depend on the user's ability to state up front requirements and the likelihood that these requirements will not change significantly with time. More specifically, the user requirements should not change in the time it takes to develop and field the new product. In cases where the user's initial requirements are uncertain, the spiral process addresses this through the use of system representations to solicit feedback on the requirements (Dare 2003). In the cases where it is likely that the user's requirements are dynamic, but not drastically enough to warrant a new development, the use of evolutionary development to evolve the final requirements areas is logical as well. Program B also serves as the case with perhaps the most

uncertainty, as it was implementing new business rules and provided new capabilities. This program also had a very high level of iteration and the most spiral-like approach of the programs researched. Additionally, Program B, in which the first increment was a significant leap forward over previous capabilities, also had a long increment, on the order of five years. This is an interesting find, as many expect spiral development to shorten the development time. In fact, spiral development does not necessarily do this, but ensures that in the given time the user is provided with a system that best suits their needs.

A third concern is timeliness. If the user requires a technology immediately, a long, drawn-out process does not make sense. At the same time, if the products targeted environment or user requirements are changing, then the product is most useful if it delivered before these changes take place. Essentially, it is crucial to match the speed of the program with the rate of change of the factors that product is to address. Using an incremental strategy tends to provide the most frequent updates to an end user, while a waterfall process or the Boehm Spiral does not necessarily imply rapid deliveries. Accordingly, the program must choose a process is appropriate to the needs of the user and the operational environment. For the majority of programs studied, both technical and user requirements were relatively certain. In these programs, the goal was to provide the user with capability as quickly as possible. The result was an incremental strategy much more along the lines of block upgrades than anything else. Program D exemplifies this, as it had very clear requirements and used a derivative of existing technology. This was matched by an incremental strategy that was primarily defined up front and users were given new systems as quickly as the program could provide them.

Table 4 summarizes the conclusions of this research about the use of various development strategies.

Table 4: Notional diagram of processes and uncertainties

	Requirements uncertainty	
	High	Low
Technical uncertainty: High		Waterfall
Technical uncertainty: Low	Spiral	Incremental

This research found that in cases of high technical uncertainty and low requirements uncertainty, there was a tendency towards a more traditional approach. In programs with high requirements uncertainty and low technical uncertainty, a more iterative process such as spiral development was found to be more appropriate. Lastly, in those programs where the uncertainties are minimal, and the goal of the program is to rapidly provide the user new capabilities, an incremental process using a block upgrade strategy seems to work best. This should not be surprising, as this had historically been the goal of the block upgrade methodology identified in Chapter 3, and evidenced in the F-16 program. In the F-16 case study, it was shown that the block upgrade process provided the program with capabilities very quickly, allowing it to address various user needs. In fact, one could argue that historically, the block upgrade process emerged as a strategy to handle evolving needs.

7.2 Applicability of Evolutionary Acquisition

Though Evolutionary Acquisition has become the mandated acquisition strategy in the Air Force, this does not mean that it always makes sense. As discussed above, in high-risk technical environments, an evolutionary process does not necessarily provide capability to the user in the

shortest time. In many cases evolution is not possible on a rapid timescale without an existing platform—particularly a platform with evolvability through system modularity. Platforms, especially in airframes and satellites and large, complex systems typically have greater risks and take longer to develop. In many cases, a longer process such as a spiral development or waterfall best serves platform development. As a result, Evolutionary Acquisition does not always apply. In the case of the F-16, for example, it took approximately nine years from the request for proposal to initial operational capability. Subsequent blocks, however, took only about five years, a reduction in cycle time of nearly fifty percent. Research by Beckert (2000) indicated that derivatives of existing platforms take approximately a third the time of the platform development. Historically, as the level of cutting-edge technology has risen, there has been a distinct rise in the development time (as shown in Figure 1). In Programs B and F, the first increment took or is expected to take nearly five years. Likewise, the platform development for Program E took much longer than the spirals the program is currently working on.

With Evolutionary Acquisition, the goals are two-fold: 1) deliver capability to the user quickly, and 2) afford the best possible system to the user. For the most part, the case studies interviewed were doing exactly this. They were focused on rapidly delivering new systems and developing future increments based on these. Some programs used demonstrators and prototypes in this process as well. Program D, which had very fixed requirements and schedule pressure, implemented a concurrent, incremental development to rapidly address the user needs. In programs that provided new capabilities that were not like prior systems, such as Program B, the process was much more iterative, but the first increment took some time, and successive increments cost more than anticipated.

In Chapter 3 a variety of product development processes were identified. Of note were the incremental and spiral development processes, on which Evolutionary Acquisition is based. The premises for the two were radically different—incremental delivery is focused on delivering capabilities over multiple stages, while the Boehm model of spiral development focuses on delivering a single capability through user feedback based on a series of evolving system representations. The Air Force's plan for Evolutionary Acquisition is somewhere in between the two, an incremental strategy in which future deliveries are based on feedback from current systems.

For Evolutionary Acquisition—the strategic and controlled evolution of systems to quickly adapt to changing needs—to work effectively, there must be a number of innate attributes to the system and environment.

- The operational environment must not be one in which there is a significant level of major operational requirement or environmental shifts, or in which there are significant disruptive technologies. This is typically not in the control of the program, but programs must be aware of this in selecting a development strategy. That is to say that programs should be aware that in abruptly shifting system environments it is difficult to adapt existing systems to new needs. Program B, for example, has had a number of challenges due to the fact that the operational environment is in a state of flux, and the program has had to cope with these changes. That is to say that a radical shift in the operational environment or the need for new capability might best be served with a new platform. Program B is one example of this, where it represented a significant new capability. The

F-16 development itself is one example. A fundamental shift in war strategies led to a need for a low cost fighter—a system that could not have been evolved from an existing aircraft.

- In a dynamic environment, the system must be able to evolve at a rate on part with the environment. This is particularly true in systems of systems or products that have a great deal of interdependencies. The same can be said of user needs—where these are subject to change, it is important that the system be able to keep up with this. The F-16 case study is one example of this, as is Program D. Specifically, Program D was able to adapt to changes in the operational environment through its incremental delivery process in a timely manner. The F-16 is a stronger example, as various block upgrades, specifically night and all-weather upgrades, have transformed the role of the F-16 from lightweight daytime fighter to strategic bomber.

7.3 Implementing Evolutionary Acquisition

Assuming the program has decided to use Evolutionary Acquisition, there are a number of activities the office should perform to efficiently execute the strategy. Moreover, if the Air Force sees Evolutionary Acquisition as it's prime strategy for acquiring new capabilities, then Air Force must also implement changes to better address the challenges of this group of processes.

Program implementation

Program management is difficult regardless of the process. In an evolving system, this management challenge is even greater. The research identified a number of key challenges and corresponding practices that program managers used to mitigate these challenges. The recommendations in this section specifically look at how program managers can manage

evolutionary programs in the current environment. That is to say that several of these recommendations would be best addressed not by program managers, but rather by policy changes in the supporting organization.

Operational requirements

- *Recommendation:* Programs either need funding flexibility to address new requirements or a user community that understands the role of requirements in Evolutionary Acquisition—i.e. that requirements can be fed into future increments.
- *Findings:* While operational requirements will inevitably change throughout the development process, program managers used two strategies to reduce changes. These were 1) prioritizing requirements with the user and 2) allowing requirements changes only when additional funding is provided. This seemingly runs contrary to the goals of Evolutionary Acquisition—providing the user with a system that best addresses their latest needs. In other words, program managers tend to want to freeze requirements early on so as to better plan the process and execute to a predictable schedule. Accordingly, it seems more logical to provide a management reserve or slush fund for addressing requirements changes that provide the user with significant capability. The majority of program managers were primarily budget constrained in making changes, leading to the tendency to freeze requirements. This was specifically cited by Program B and implied in a number of programs. Another option is to better educate stakeholders—especially the user—as to what to expect in evolutionary programs (further detailed under the bullet "managing stakeholder expectations").

Technology obsolescence

- *Recommendation:* In order to evolve in general, and more specifically to avoid technology obsolescence, programs must control the interfaces between subsystems and modules. Additionally, programs should use standardized off-the-shelf components only after serious consideration and a lifecycle analysis.
- *Findings:* Typically this problem was addressed with the use of off-the-shelf technologies, but as the program managers in B and D noted, this provided a bigger challenge in the long run as commercial products obsolesced more quickly. In fact, both indicated that previous military standard systems were much less susceptible to obsolescence. Furthermore, the process is aggravated by the inability to control the interfaces of these systems. This is actually the key point, and interface definition is the key to open architecture systems.

Budgeting

- *Recommendation:* Multiple sources of uncertainty create greater management challenges. If the goal of Evolutionary Acquisition is to afford users the best possible system in a given time, program managers must be able to execute programs to plan—this means a static, predictable funding profile.
- *Findings:* program managers often faced budget cuts and were unable to keep reserves, so the result was that they used the requirements as reserves and cut requirements accordingly. The fundamental challenge in a dynamic program is that the funding is not predictable. This is a necessity for Evolutionary Acquisition.

Open architectures

- *Recommendation:* To rapidly integrate new capabilities, programs must own the interfaces and data rights between modules and subsystems. Specifically, this means two

things: 1) develop their own standards based on commercial standards or otherwise, as was the case with military standards historically, and 2) purchase data rights from commercial companies so that the Air Force owns and operates the data transfer between systems, which is crucial for integration.

- *Findings:* It is imperative for evolving systems to be able to rapidly integrate new capabilities and features. In the case of the F-16, this was done through the use of open architectures. The key here is that programs own data rights in the case they use commercial products, and otherwise control the interfaces.

Minimize interdependencies:

- *Recommendation:* Programs that must evolve their compatibility with external systems must either evolve at the rate that the operational environment does, or alternatively decouple themselves from the operational environment. This can be done in a couple of ways. The first is to implement an open architecture-like strategy in a system of systems, such that systems are interoperable through some type of controlled interface, and therefore individual systems are free to some extent to evolve separately. The second strategy is to match the evolution of systems to their operational environment by defining these changes in advance and allowing programs to implement a new standard by a given date. This is very much the case for Program D for example.
- *Findings:* As Figure 29 showed, the challenge to making changes in programs tended not to be the system itself so much as the interdependencies it had with external systems. As a result, it becomes important to reduce these. Where that is not possible, the goal should be to match the cycle time of development—i.e. the iteration or increment time—to the rate of evolution of the external system.

Resource planning:

- *Recommendation:* In executing an evolutionary strategy, programs must expect that resource demands will be higher. Programs should either plan for additional resources or use a development strategy that better suits available resources, i.e. a less iterative strategy. In other words, programs must either hire additional staff members to deal with contracting and planning, which are within the realm of the program office. In areas such as logistics and users, programs must make an effort to further involve these groups through collocation or alternatively provide greater incentive for participation. In the area of testing, while beneficial to bring in the test community early for planning the development strategy, as was the case with Program D, it is also valuable to invest in a testing infrastructure for the program, and procure what test assets and facilities are possible and necessary. It would also be very valuable to programs to use process improvement tools and methodologies such as lean to maximize the efficiency of given resources.
- *Findings:* Many of the programs experienced or anticipated a number of staffing resource issues. Specifically, these were in contracting, logistics, planning, testing, and user involvement. These were typically resources to deal with the acquisition system that has not necessarily changed alongside the process. Particularly, the program can be expected to have shortfalls in planning resources as increments run concurrently. Additionally, contracting, logistics, and testing were large challenges, as these occur more frequently in an evolving system. Naturally, another big need is to have greater user involvement. The fundamental challenge is that the communities outside of the program offices have not changed their practices to facilitate Evolutionary Acquisition.

Mature technologies:

- *Recommendation:* To evolve new capabilities rapidly, programs must not spend time in maturing new technologies. This task must be done external to the program, and only readied technologies should be brought in for integration. This appears to be contradictory to the reduced dependence on commercially available systems. The focus in using commercial systems (rather than military developed systems) must be in addressing the interface issues rather than the capability of the commercial system. That is to say that programs must focus most on integration rather than deriving or further developing commercial systems. Another option is to use military designed systems, as the case was for the F-117 and F-18, both of which used a derivative of the F-16's fly-by-wire system.
- *Findings:* The use of mature technologies is fundamental to Evolutionary Acquisition. As the goal of this strategy is to rapidly integrate new capabilities and provide them to the user, the program should not be burdened with the job of developing and maturing technologies. This was not a problem in these programs, as for the most part they used mature technologies. Nevertheless, some programs cited this as a fundamental part of evolutionary strategies.

Managing stakeholder expectations:

- *Recommendation:* In order to maximize program flexibility and manage stakeholder expectations, organizations must involve the user either by using system representations or educating the user through the use of internal agents. This essentially is providing the program with a better path for communication with the user. System representations provide valuable information to the user in a tangible, experiential way, giving the user a way of comparing the development to their expectations. The use of agents within the

user community is a means for the program to feed information into a handful of disseminators, who essentially serve as a translator from the program to the users. At the same time, these figures can also educate the users from within, where they are more likely to change.

- *Findings:* Several program managers' biggest challenges had to do with the user's ability to understand the acquisition strategy and its implications for the user. Users would then request that many requirements be addressed in the first increment, leading to a waterfall-like program, or would be dissatisfied with the incremental process, and not buy in to Evolutionary Acquisition. Program A, which had a user with spiral experience, did not have issues with a failure to understand the role of increments. Program managers addressed this in two ways. The first of these was the use models, prototypes, and demonstrators, to provide the users with an idea of the system capabilities and an opportunity to gather feedback and evolve requirements. The second strategy was to develop agents within the user community that understand the process and were embraced by the program office as disseminators of information.

Policy recommendations

Though programs can do a lot to help the likelihood of success in Evolutionary Acquisition, a number of things extend beyond the control of the program office. Specifically, the challenge is that while the acquisition community has pushed for Evolutionary Acquisition, government policy and other groups within the defense community have not necessarily embraced it. The fundamental challenge is that these groups are not all on the same page.

Testing

- *Recommendation:* The testing process must be updated to apply to evolving systems. Full scale testing for each increment or deliverable is not necessarily practical in systems with a high level of iteration. At the same time, it is imperative that these systems perform safely and suitably. Full scale testing should be performed for the platform, as this will serve as the basis for future evolution. Beyond this, it is not necessary to retest all systems for each increment. The manager for Program B noted that the tests were repetitive, as did one of the managers for Program C. Accordingly, there are two things that are necessary in the testing community to address iterative developments. The first of these is to 'lean' out the process, using lean methodology to perform testing more efficiently and more quickly. This will reduce both the testing resources necessary as well as the test time. The second part is to specifically address changes in the system rather than retest the system. This applies well to modular systems, as essentially only those subsystems upgraded or added should require full testing. This suggestion of regressive testing was put forth by one of the managers in Program B.
- *Findings:* Testing is more frequent in evolutionary programs, and accordingly the traditional system of testing is not practical for a highly iterative system. Though involving the test community is one way of improving this, it is also necessary to foster a test environment that matches the rapid cycles of Evolutionary Acquisition.

Logistics

- *Recommendation:* Logistics will undoubtedly be a challenge in programs delivering multiple capabilities over time. One need look no further than Program E for an example of the challenges in developing a number of blocks or lots. The F-16 program did this in a couple of ways. First of all, many of the systems were designed for maintainability and

modularity. This allows for ease of maintenance and logistics. It also allows for simplifying the upgrade process, which is the second, and perhaps more important strategy, of having a series of retrofit programs to bring previous generations up to new standards where possible. This will be done in Program E as well. This must be planned for during platform development, and hence it is important to involve the logistics community in the development process. Retrofitting does not make sense every generation, due to excessive costs as well as the fact that this takes out valuable inventory from active use. A more likely scenario is the midlife upgrade strategy that is currently practiced throughout the military. Furthermore, logistics must play a role in planning for and addressing the reality of support in an evolutionary system. As more than one program manager said, logisticians were involved in teams, but were dragged along, kicking and screaming. Another challenge arises if the program is highly experimental—that is to say that it is not producing significant volume. This is the case of a number of ACTD programs that provide a useful capability though only a handful of systems are available. This also provides an opportunity for use and evaluation in a real-world environment. One potential way to handle this is to establish a reserve fund for such systems, such as suggested in the budgeting section below.

- *Findings:* Logistics is a further challenge for these incremental systems. Again, logisticians must be brought into the development process, but there is a fundamental challenge in releasing multiple increments and supporting them. The best way to minimize this challenge may be to bring prior system up to the latest standards, as was the case with the F-16 program.

Budgeting

- *Recommendations:* Funding continues to be a challenge in Evolutionary Acquisition. This is primarily because incremental programs are likely targets of funding cuts, given that systems already exist, and requirements can be pushed into future increments. Nevertheless, work within the Air Force as well as this research indicates that a stable funding profile is very beneficial to Evolutionary Acquisition. Program C, which was given fixed funds to deliver a new capability is one such example. Program B, suffering from a number of financial challenges is struggling with future increments. There are a number of solutions to this. The first of these, suggested by many including Spaulding (2003), is to establish a funding wedge or slush fund for evolutionary programs. This money can be given to programs to address cycle time issues as well as unforeseen circumstances in evolving systems. This also addresses programs' inability to accept new requirements. If changes to requirements are deemed sufficiently valuable, additional funding could be provided to address these changes. This fund could be owned by a number of sources, depending on the level of implementation. In order to completely back evolutionary programs this could be a line item in the Air Force budget, or alternatively funding could be managed at the system director level to be allocated to various programs within the director's portfolio. The advantage of having an Air Force line item budget is that the Air Force could also back 'pure' spiral development, funding programs like Program B during a phase between one increment and the next, as illustrated in further detail by Spaulding's work. This would not only maintain the program office, but allow for greater use and evaluation feedback for future increments. Harking back to a much more traditional system, program managers could be allowed to keep a reserve. Though this has fallen out of favor in recent years, a budget reserve

would allow programs to deal with unforeseen circumstances. Regardless, it is evident that for highly iterative and evolutionary programs—especially if the user is to be provided the most suitable capabilities—a change in the funding system is imperative.

- *Findings:* Budgeting proved to be a fundamental challenge to many programs in several ways. First of all, due to the uncertainty and flexibility of incremental programs, funding must be stable, and programs management must be allowed to keep a reserve for unforeseen situations.

Education

- *Recommendation:* A crucial part of the development process in Evolutionary Acquisition is the strategic goal of delivering incremental capability improvements that afford the user with the best possible capability. This requires that the user both be highly involved in the process and that they understand that each delivery is not the final solution. Programs used two means to educate and inform their users—system representations and change agents, as stated above. Nevertheless, if Evolutionary Acquisition is to be successfully implemented across the Air Force, the burden of education is better placed on the Air Force. There are several possibilities for improving user involvement and awareness. The first of these is an educational program, like training, for users. Though this can teach some of the lessons necessary in Evolutionary Acquisition, it does not necessarily better involve the user. Providing the user incentive to participate does this more effectively. One such strategy is the use of system representations, which are more 'on the level' of the user. Moreover, collocating a small user group with the program (or the program office with the user) would provide ease of access and facilitate communication between these two key players. The most effective strategy is likely a

combination of the three—brief training for users to provide an overview of the strengths and weaknesses of evolutionary processes, and the use of system representations along with collocation to improve collaboration among the two major stakeholders.

- *Findings:* Stakeholder education is key in evolutionary programs. A number of programs noted that users did not fully understand the implications of an evolutionary process. This included a failure to comprehend the level of user involvement required, the resources required, and what deliverables the user would receive. More fortunate programs, such as Program A, had users that were familiar with spiral development, who understood the user role and were aware of the tradeoffs with such programs.

Military standards

- *Recommendation:* Return to military standards. Though this is no longer a popular strategy, and may be somewhat controversial, the strategic use of military standards historically has helped the Air Force to control and own the interfaces between subsystems in a system. In the future of systems of systems, such a strategy will help to maintain interfaces across multiple systems. This deals directly with the challenges of off-the-shelf technologies, where control is outside the hands of the Air Force. Furthermore, this helps to address many of the logistics and upgrade issues as programs evolve. Controlling these interfaces will not only allow programs to evolve rapidly, but also to reduce the friction caused by disparate rates of evolution across multiple subsystems and systems. Though initial development costs may rise, the support and maintenance costs should drop, and the need for technology refreshes to address diminishing manufacturing sources should be reduced. The fundamental recommendation of this research is not that military standards should replace the use of

commercially available systems, but that the policy should incorporate a lifecycle analysis of costs in an evolutionary system to determine when commercial systems are appropriate. In cases where these are selected, the Air Force must ensure that they procure data and interface rights, so as to maintain the open architecture standards.

- *Findings:* Programs C and D both noted that the use of off-the-shelf components did not necessarily help the program for different reasons. In Program C, this was because the use of these systems required frequent technology refreshes which not only meant increased costs, but increased development work by the program. In Program D, the systems that were available did not necessarily suit the operational environment of the program. Additionally, Program A, also using off-the-shelf components was also planning for a number of hardware refreshes as well. That is not to say that the use of these components is bad, but rather that an analysis of lifecycle costs could indicate whether the use of these systems actually saved money in the system lifetime.

7.4 The future of Evolutionary Acquisition research

Though this research highlighted a number of findings and made key recommendations for implementing Evolutionary Acquisition strategies, it did not satisfactorily answer the question of when various development strategies make sense. Some of this will be answered in a forthcoming paper from the Lean Aerospace Initiative, but much of it a potential source for future research. In the literature review in Chapter 2, the work by Brown and Eisenhardt indicated that there were a number of organizational issues to consider beyond the product and uncertainties. The recommendations in this last section themselves indicate a number of areas for future research. In particular, the macro issues of testing, logistics, and budgeting are beyond the scope of this research, but nevertheless crucial to the product development process.

- The role of testing in Evolutionary Acquisition. This involves two things—the first a study of how to best implement recurring testing for these systems, and the second to better understand how to lean out testing in general. In doing this there are a several of key points that might be addressed. One proposal by a program manager was to perform regressive testing. While doing this could significantly reduce testing times, implementing this might be very difficult, especially as system functions are rarely completely decoupled. Research in how to ensure that the safety and mission-critical aspects of the system are adequately addressed in a regressive testing strategy would be very valuable. From a slightly more systems engineering perspective, design for manufacturing has been a huge success, but design for testing is unheard of. Nevertheless, as testing is necessary, it makes sense to address these issues in the design and development of new products. A third potential research topic is a comparison of test groups within the program to testers as external stakeholders—programs had good success with owning their own test assets and facilities, but the test community itself still acts as an outside force on the system, as was the case in Program D, where the test community wanted greater control of testing after the LRIP.
- The ability to develop system representations. Though system representations are valuable to the program, this is not always easily done. In many cases the cost of these representations is prohibitive—particularly if the program intends to prototype frequently as in a spiral development strategy. Much more needs to be learned about where to strategically develop representations, such as in the human-machine interfaces, performance models, and so forth to provide the most valuable feedback from the user at the lowest possible cost to the program.

- A value comparison of development strategies to various stakeholders. This research only begins to uncover some of the various value propositions among the different product development processes. In particular, this research fundamentally looked only at the user side value proposition, but did not necessarily look at how each of these strategies affects other stakeholders.
- Organizational and cultural challenges in Evolutionary Acquisition. In the early stages of this research, the author looked at organizational roles in different development strategies, but realized that this broadened scope of the research too much. Nevertheless, it is likely that fundamentally different strategies would best suit different organizational structures, or that conversely to implement certain strategies organizations could be arranged differently. In particular, this implies that if the Air Force is to implement Evolutionary Acquisition, it may be better served by addressing cultural and organizational transformation as well.
- From a technical perspective, more research is necessary on the role of modularity and open architectures in evolving systems. The result of this research indicated a low correlation between ease of changes and modularity. This should also include the use of non-developmental and commercially available systems.
- The budgeting process is an essential part of this picture, and though there are some clear recommendations on how to address evolutionary acquisition, the potential for greater understanding of this system is necessary. In particular, this research should look at the level of funding allocation, and when additional funding to implement new capabilities or address requirements changes is necessary or valuable.

In concluding, the work presented makes a number of recommendations based on the findings of the case studies. Nevertheless, there are a number of research questions that are highly applicable to programs intending to implement Evolutionary Acquisition that are left unanswered by this work. In short, while this research provides guidance to program managers and policy makers on product development strategies and the role of evolving systems, it represents only the beginning of research in this area.

Appendix A - Surveys

Program Management Survey

This survey is part of continuing work by the Lean Aerospace Initiative at MIT, in support of the aerospace community and the Air Force. In particular this research is investigating evolutionary acquisition strategies and product development processes, such as waterfall, incremental, and spiral development, and how they apply to various programs.

Please answer the following questions as they apply specifically to your individual program or project. If your program is part of a larger program office, DO NOT answer the questions as they might apply to that larger organization, but rather your specific project and task. In addition to those questions, the survey refers to questions on the product or system. This system refers specifically to the system being developed by your program—whether it is a subsystem, module, or software, or even a system in a system-of-systems, please consider the specific system that is being developed in your program. In addition please consider only the current program phase and contract.

Please be candid and honest in your responses. We understand that you may have concerns about confidentiality. Several measures will be taken to ensure that your responses will remain confidential. All analysis of the survey data will be presented in the form of aggregated statistics. No individuals or individual programs will be identified in the analysis or reporting of the responses. You may decline to answer any questions, and are not obligated to continue participation. We understand that the success of any research depends upon the quality of the information on which it is based, and we take seriously our responsibility to ensure that any information you entrust to us will be protected with the same care that you would use yourself.

The survey has been designed into several sections identifying several key categorical attributes and processes. It has been designed to take approximately 40 minutes.

PROGRAM BACKGROUND

1.Program title/project: _____

2.Office symbol: _____

3.Program length to date (in months): _____ mo.

4.Estimated acquisition program baseline budget: _____

5.Are you using an evolutionary acquisition strategy, where system capabilities are delivered incrementally?

☐ Yes ☐ No

6.How far along in the program would you estimate you are: ____% complete

7.Please select the category below that best describes the type of system your program is developing:

☐ Aircraft (airframe and mechanical systems)
☐ Aircraft (avionics and electronic systems)
☐ Aircraft (propulsion)
☐ Spacecraft or launch system
☐ Electronic system (non-aircraft)
☐ Missile or munitions
☐ Software-dominated system
☐ Other: _____

SPIRAL CHAIN

1.Is the program using spiral development?

☐ Yes ☐ No *(skip to next section)*

2.How many total spirals are currently planned for in the acquisition strategy (indicate number)? _____

3.What spiral are you currently in (indicate number)? _____

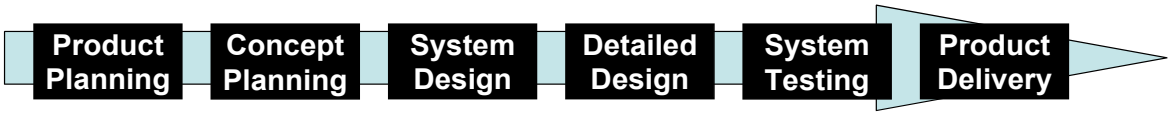
4.Estimate the resource needs in the following categories in your program versus a non-spiral or traditional acquisition program of similar magnitude (circle the value that best describes your program):

	Less			No Change			More
Staffing (Administration)	1	2	3	4	5	6	7
Contract Oversight (PCO / Buyer)	1	2	3	4	5	6	7
Financial (Cost Estimate / Budget)	1	2	3	4	5	6	7
Planning (Program Management)	1	2	3	4	5	6	7
System testing	1	2	3	4	5	6	7
Coordination with other offices / agencies	1	2	3	4	5	6	7
Engineering	1	2	3	4	5	6	7
Logistics	1	2	3	4	5	6	7
User or Testers in collocated in program office	1	2	3	4	5	6	7

5.To what extent has the user committed to up-front funding necessary for capability demonstration?

User has not been committed/prefers traditional funding profile 1 2 3 4 5 6 7 User has been highly committed/agreed to up-front funding profile

PROCESS IDENTIFICATION



This image depicts the major activities in a traditional product development process. In a traditional waterfall process for example, development activities are sequentially executed, and passing a "stage-gate" or milestone allows the program to continue to the next activity. Additionally, these activities are not intentionally revisited once the phase is deemed complete in a stage-gate review.

CHECK THE BOX BELOW THE OPTION THAT BEST DESCRIBES YOUR PROGRAM

Please consider this image and text when answering the following two questions:

Program milestones serve as stage-gates to finalize one development activity (design, etc.) and proceed to the next activity with little to no interaction between activities ☐	Program milestones overlap or are intended to loosely evaluate the program progress and there is a high degree of interaction between activities ☐

Major development activities (as shown above) are intended to be executed only once in the program ☐	Major development activities (as shown above) are intentionally revisited or repeated multiple times in the program ☐

Capabilities are provided through:

A single well-defined system or deliverable ☐ Skip next question	Multiple incremental systems established by initial requirements document ☐	Multiple incremental systems, based on use and evaluation of prior deliveries ☐

Evaluate the mechanism for requirements changes between successive deliveries (circle the value that best describes your program):

Requirements evolved from strategy identified early in program — 1 2 3 4 5 6 7 — Requirements evolved by feedback from user use and evaluation

Initial delivered capability is:

the full operational capability (FOC) established by requirements document ☐	the full operational capability (FOC) established through requirements changes during prototype evaluation ☐
the first in a series of deliveries with requirements primarily defined up front for successive capabilities ☐	the first in a series of deliveries with requirements for successive capabilities emerging from use and evaluation ☐

How strongly are performance requirements affected by schedule or budget constraints (circle the value that best describes your program)?

Cost / schedule are the primary requirements drivers — 1 2 3 4 5 6 7 — Cost / schedule are not major requirements drivers

Has the acquisition strategy used to develop the system changed dramatically during the project (i.e. shift from traditional to spiral development, etc.)?

☐ Yes ☐ No

If yes, please explain:

PRACTICES

1.Evaluate the level of impact of the following groups in developing system requirements (circle the value that best describes your program):

Not involved beyond initial requirements			Occasionally involved in design reviews, etc.			Heavily involved on product team, etc.	
End user	1	2	3	4	5	6	7
Wargames / Experimentation	1	2	3	4	5	6	7
Modeling & Simulation	1	2	3	4	5	6	7
Testers	1	2	3	4	5	6	7

2.Evaluate the level of warfighter experience with your system (circle the value that best describes your program):

No experience with system / experience with highly dissimilar systems			Experience with somewhat similar system			Experience with very similar system or different version of system
1	2	3	4	5	6	7

3.Is / was an analysis of upgrades through the system's lifetime performed?

☐ Yes ☐ No

4.Are simulations and modeling employed in the development process?

☐ Yes ☐ No

5.Are these models and simulations used to engage the user in evolving and verifying requirements?

☐ Yes ☐ No

6.Are models and simulations intended to be updated and evolved for redesign or successive designs?

☐ Yes ☐ No

7.Does the program have the ability to rapidly produce prototypes or perform builds at the system level?

☐ Yes ☐ No

8.Are prototypes used to engage the user in evolving the final deliverable?

☐ Yes ☐ No

9.If the product is part of a larger product (i.e. a subsystem in a system, or a part of a system of systems), how often is the integrated system tested (select one)?

1. Never
2. Once (at program completion)
3. At major milestones
4. In regularly scheduled events
5. Other: _____

10.Evaluate the flexibility of changes to the design of the system through prototype delivery and evaluation (circle the value that best describes your program):

Design changes difficult to evolve			Some design changes possible			Design changes easily integrated
1	2	3	4	5	6	7

11.What percent of activities in the test plan (TEMP) are satisfied through combined developmental and operational testing and evaluation? _____%

PRODUCT INTENT

1.Is the product being developed as an improvement / derivative of an existing product?

☐ Yes ☐ No

2.Does the acquisition strategy imply or intend that the product's performance or capability is to improve or change with time?

☐ Yes ☐ No

3.Evaluate the impact of external constraints (i.e. interoperability, requirements on use of specific technologies, etc.) on the key design decisions (circle the value that best describes your program):

External constraints did not significantly impact key design decisions			External constraints impacted some key design decisions			External constraints dominated key design decisions
1	2	3	4	5	6	7

TECHNOLOGY

1.Level of technology advances required in components critical to meeting program technical objectives (circle the value that best describes your program):

All non-developmental items (COTS, etc.)			Some development required			All new development required
1	2	3	4	5	6	7

2.Indicate the percent of recurring non-developmental, sometimes referred to as "fly-away", costs (in the case of software, total developmental costs) attributable to purchase of off-the-shelf (OTS/COTS) or non-developmental (NDI) technologies: _____%

3.Evaluate the rate of changes of the technologies underlying this system's primary functional performance (circle the value that best describes your program):

Technologies evolve more slowly than product			Technologies and product evolve at similar rate			Technologies evolve more rapidly than product
1	2	3	4	5	6	7

ARCHITECTURE

1.Evaluate the level of interface definitions in subsystems or modules that had to be developed specifically for your project: ___%

2.Evaluate the ease of changes in subsystems or modules (circle the value that best describes your program):

Subsystems or modules are tightly integrated into the system, and significant modeling or testing must occur for changes or new systems						Subsystems or modules are highly modular, with minimal modeling and testing necessary for changes or new systems
1	2	3	4	5	6	7

3.Indicate the total percent of all requirement changes that involved changes in software requirements: ____ %

4.Evaluate the level of key functionality implemented through software (circle the value that best describes your program):

Functionality requires no software implementation	1	2	3	4	5	6	7	Functionality relies fully on software implementation

UNCERTAINTY

1.Evaluate the following in terms of the uncertainties experienced to date in the product development process (circle the value that best describes your program):

	Unlikely						Likely
Actual funding matched requested funding	1	2	3	4	5	6	7
Operational requirements remained constant	1	2	3	4	5	6	7
Technical advances necessary for program success were achieved	1	2	3	4	5	6	7
Parts or technology obsolescence	1	2	3	4	5	6	7
Operational environment remained the same	1	2	3	4	5	6	7

2.Evaluate the following in terms of the uncertainties *throughout the product development process* (circle the value that best describes your program):

	Unlikely						Likely
Actual funding matches requested funding in the future	1	2	3	4	5	6	7
Operational requirements remain constant	1	2	3	4	5	6	7
Likelihood that technical advances necessary for program success are achieved	1	2	3	4	5	6	7
Likelihood of parts or technology obsolescence	1	2	3	4	5	6	7
Operational environment remains the same	1	2	3	4	5	6	7

3.Evaluate the adaptability of your *SYSTEM ARCHITECTURE* to external changes such as the threat, funding, or user requirements changes (circle the value that best describes your program):

System architecture easy to adapt 1 2 3 4 5 6 7 System architecture difficult to adapt

4.Evaluate the sensitivity of overall system *performance* to the inability of a key subsystem to achieve intended performance (circle the value that best applies):

Not sensitive 1 2 3 4 5 6 7 Highly sensitive

5.Please estimate your program's cost change to date if it exists. Calculate this as the percentage change in actual expenditures compared with *initially planned* expenditures for this point in your program's progress, using actual-year dollars. Treat increases beyond original plan as positive cost change.

Total program cost change _____ (+/– % to-date)

Indicate baseline year _____

6.Please estimate your program's schedule variation if one exists. Calculate this as the difference between your current schedule and the initial program (treat increases beyond original plan as positive schedule variation).

Total program schedule variation to date _____ (+/– months to-date)

7.To what extent has (is) technology maturation required additional development within your project?

Significant development has been required to mature technology	1	2	3	4	5	6	7	Insignificant or no development has been required to mature technology

8.Where have you had the greatest challenges in executing evolutionary acquisition strategies?

_____

_____

_____

System Program Director Survey

This survey is part of continuing work by the Lean Aerospace Initiative at MIT, in support of the aerospace community and the Air Force. In particular this research is investigating lifecycle processes and various product development processes, such as waterfall and spiral development, as they apply to various programs. The survey attempts to identify key overarching issues in acquisition strategies at the program office level.

Please answer the following questions as they apply to the group of programs within your program office.

Please be candid and honest in your responses. We understand that you may have concerns about confidentiality. Several measures will be taken to ensure that your responses will remain confidential. All analysis of the survey data will be presented in the form of aggregated statistics. No individuals or individual programs will be identified in the analysis or reporting of the responses. You may decline to answer any questions, and are not obligated to continue participation. We understand that the success of any research depends upon the quality of the information on which it is based, and we take seriously our responsibility to ensure that any information you entrust to us will be protected with the same care that you would use yourself.

The survey has been designed into several sections identifying several key categorical attributes and processes. It has been designed to take approximately 10 minutes.

1.Office symbol: _____

2.Does the acquisition strategy specifically define that the system's capability is to evolve with time?

☐ Yes ☐ No ☐ Does not apply

3.Indicate the percentage of major subsystems or projects (including software) by cost using the following development strategies (total should equal 100%):

Spiral Development: Desired capability is identified, but end-state requirements are not known at program initiation. Requirements for future increments depend upon technology maturation and user feedback from initial increments	**Incremental Development**: End-state requirement is known, and requirement will be met over time in several increments	**Traditional Development**: A single delivery in which the final capability is delivered based on up-front requirements
Number of projects ____	*Number of projects ____*	*Number of projects ____*
____% of total acquisition cost for projects	*____% of total acquisition cost for projects*	*____% of total acquisition cost for projects*

4.Evaluate the resource needs of spiral projects in your program versus the traditional projects in your group of acquisition programs of similar magnitude (circle the value that best describes your program):

	Less			No Change			More
Staffing (Administration)	1	2	3	4	5	6	7
Contract Oversight (PCO / Buyer)	1	2	3	4	5	6	7
Financial (Cost Estimate / Budget)	1	2	3	4	5	6	7
Planning (Program Management)	1	2	3	4	5	6	7
Testing	1	2	3	4	5	6	7
Coordination with other offices / agencies	1	2	3	4	5	6	7
Engineering	1	2	3	4	5	6	7
Logistics	1	2	3	4	5	6	7
User or Testers collocated in Program Office	1	2	3	4	5	6	7

5.For each development strategy, indicate the number of programs that used modeling and simulation and/or prototyping to engage the user for requirement verification and evolution?

Spiral Development	**Incremental Development**	**Traditional Development**

6.In each of the following categories, what percent of programs' test plans (TEMP) were/are satisfied through combined developmental and operational testing and evaluation?

Spiral Development **%**	**Incremental Development** **%**	**Traditional Development** **%**

7.Evaluate the certainty of operational requirements in for your programs in your program office within the following categories:

	Requirements clear, no changes from initial requirements						Requirements unclear, significant changes from anticipated requirements
Spiral Development	1	2	3	4	5	6	7
Incremental Development	1	2	3	4	5	6	7
Traditional Development	1	2	3	4	5	6	7

8.Evaluate the difficulty of executing programs to schedule for the following categories of programs:

	Easy						Difficult
Spiral Development	1	2	3	4	5	6	7
Incremental Development	1	2	3	4	5	6	7
Traditional Development	1	2	3	4	5	6	7

8.Evaluate the difficulty of executing programs to cost/budget for the following categories of programs:

	Easy						Difficult
Spiral Development	1	2	3	4	5	6	7
Incremental Development	1	2	3	4	5	6	7
Traditional Development	1	2	3	4	5	6	7

8.Evaluate the difficulty of executing programs to performance goals for the following categories of programs:

	Easy						Difficult
Spiral Development	1	2	3	4	5	6	7
Incremental Development	1	2	3	4	5	6	7
Traditional Development	1	2	3	4	5	6	7

9.Where have you had the greatest challenges in executing evolutionary acquisition strategies?

Appendix B – Case study questionnaire

1.Program title/project:

2.Program length to date:

3.Percent of program complete:

4.Estimated acquisition baseline budget:

5.Project/product type:

6.Draw a diagram of the acquisition strategy with deliveries (increments, spirals, etc.)

7. Where did you have greater or lesser resource needs versus traditional acquisition, and what were the tradeoffs?

8.Where have you had the greatest challenges in executing evolutionary acquisition strategies?

9.How much effort was spent on maturing technologies?

10.To what extent did the user agree and understand the differences in a spiral vs. a traditional program, particularly in terms of funding?

11.What issues were there with planning and budgeting?

12.What issues were there with planning test assets and range availability?

13.How were key stakeholders involved in teams?

14.Difficulty of executing with a spiral development to cost, schedule, performance.

15. Evaluate the mechanism for requirements changes between successive deliveries—how is feedback incorporated into future increments?

16.What percent of activities in the test plan (TEMP) are satisfied through combined developmental and operational testing and evaluation? _____%

17.Evaluate the following in terms of the uncertainties *throughout the product development process*:

- Operational requirements
- Technical advances
- Technology obsolescence
- Operational environment

- Budget

18.Is the product being developed as an improvement / derivative of an existing product?

19.Evaluate the impact of external constraints (i.e. interoperability, requirements on use of specific technologies, etc.) on the key design decisions:

20.Evaluate the ease of changes in subsystems or modules

21.Evaluate the level of key functionality implemented through software:

ADDITIONAL QUESTIONS

1.Evaluate the use of system representations, such as sim & mod, or prototypes. Were they used to solicit feedback from key stakeholders?

2.If the product is part of a larger product (i.e. a subsystem in a system, or a part of a system of systems), how often is the integrated system tested?

3.Does the acquisition strategy specifically define that the system's capability is to evolve with time?

4.Indicate the percent of recurring non-developmental, sometimes referred to as "fly-away", costs (in the case of software, total developmental costs) attributable to purchase of off-the-shelf (OTS/COTS) or non-developmental (NDI) technologies: _____%

5.Evaluate the flexibility of changes to the design of the system through prototype delivery and evaluation:

6.Evaluate the rate of changes of the technologies underlying this system's primary functional performance:

7.Evaluate the sensitivity of overall system *performance* to the inability of a key subsystem to achieve intended performance:

Bibliography

Baldwin C. and Clark K. 2000. *Design Rules: the power of modularity, volume 1.* Cambridge, MA. MIT Press.

Beckert, M. 2000. *Organizational Characteristics for Successful Product Line Engineering*. Master's Thesis. Massachusetts Institute of Technology.

Boehm, B. 2000. *Spiral Development: Experience, Principles, and Refinements.* Spiral Development Workshop. Carnegie Mellon Software Engineering Institute

Boehm, B. *Spiral Development: Experience, Principles, and Refinements.* Presentation, Spiral Experience Workshop, February 9, 2002. University of Southern California

Brown, S., and Eisenhardt, K. 1995. *Product Development: Past Research, Present Findings, and Future Directions.* Academy of Management Review, Vol. 20, No. 2

Capozzolli, P., et. al. 2002. 16.885 Aircraft Systems Engineering Final Report. Camrbidge, MA. Massachusetts Institute of Technology

Cusumano, M., and Nobeoka, K. 1998. *Thinking Beyond Lean*. New York, NY. The Free Press

Dare, R. 2003. *Stakeholder Collaboration in Air Force Acquisition: Adaptive Design Using System Representations*. Doctoral Thesis. Massachusetts Institute of Technology

Defense Acquisition University (DAU). 2000. *Systems Engineering Fundamentals*. Ft. Belvoir, VA. Defense Acquisition University Press.

Defense Systems Management College (DSMC). 2001. *Glossary: Defense Acquisition Acronyms and Terms*. Ft. Belvoir, VA. Defense Acquisition University Press.

Department of Defense (DoD). 2003. DoD Directive 5000.1 and DoD Instruction 5000.2.

Dougherty, D. 1992. *Interpretive Barriers to Successful Product Innovation in Large Firms*. Organization Science, Vol. 3, Issue 2

Eisenhardt, K. and Tabrizi, B., 1995. *Accelerating Adaptive Processes: Product Innovation in the Global Computer Industry.* Administrative Science Quarterly, Vol. 40

Fine, C. 1998. *Clockspeed.* Reading, MA. Perseus Books.

Forsberg, K., H. Mooz and H. Cotterman. 1996. *Visualizing Project Management.* New York, NY. John Wiley & Sons.

Henderson, R. and Clark, K. 1990. *Architectural Innovation: The Reconfiguration of Existing Product Technologies and the Failure of Established Firms*. Administrative Science Quarterly, Vol. 35, Issue 1

Highsmith, J. 2000. *Adaptive Software Development: A Collaborative Approach To Managing Complex Systems*. New York. Dorset House Publishing

Kirtley, A. 2002. *Fostering Innovation Across Supplier Networks*. Master's Thesis. Massachusetts Institute of Technology

Meyer, M., P. Tertzakian & J. Utterback. 1997. *Metrics for Managing Research and Development in the Context of the Product Family*. Management Science, Vol. 43, No. 1

Owens, J. and Cooper, R. 2001. *The Importance of a Structured New Product Development (NPD) Process: A Methodology*. The Institution of Electrical Engineers. Vol. 10

Paone, C. "Dr. Sambur addresses need for transformation" December 6, 2002. The Hansconian

Roberts, C. 2003. *Architecting Evolutionary Strategies Using Spiral Development for Space Based Radar*. Master's Thesis. Massachusetts Institute of Technology

Spaulding, T. 2003. *Budgeting for Evolutionary Acquisition and Spiral Development*. Master's Thesis. John F. Kennedy School of Government, Harvard University

Tondreult, J. 2003. *Improving the Management of System Development to Produce More Affordable Military Avionics Systems*. Master's Thesis. Massachusetts Institute of Technology

Turner, D. 2002. "Age of Nvidia" Salon.com, May 15. (URL: http://archive.salon.com/tech/feature/2002/05/15/nvidia1/print.html)

Ulrich, K. and Eppinger, S., 2000. *Product Design and Development*, Second Edition. Boston, MA. McGraw-Hill Higher Education.

Unger, D. 2003. *Product Development Process Design: Improving Development Response to Market, Technical, and Regulatory Risks*. Doctoral Thesis. Massachusetts Institute of Technology

Webster's Revised Unabridged Dictionary. 1996.

Wirthlin, Joseph Robert. 2000. Best Practices In User Needs/Requirements Generation. Master's Thesis. Massachusetts Institute of Technology

Wolfwitz, P., Deputy Secretary of Defense. 2002. Memorandum, Defense Acquisition, Office of the Deputy Secretary of Defense.

Von Hippel, E. 1988. *The Sources of Innovation.* New York, NY. Oxford University Press.